SOBRE LA MULTIPLICACIÓN ESCALAR EN CURVAS ELÍPTICAS DEFINIDAS EN CUERPOS DE EXTENSIÓN ÓPTIMO

Por

Einstein Rafael Morales Morales

Tesis sometida en cumplimiento parcial de los requerimientos para el grado de

MAESTRÍA EN CIENCIAS

en

MATEMÁTICAS (COMPUTACIÓN CIENTÍFICA)

UNIVERSIDAD DE PUERTO RICO
RECINTO UNIVERSITARIO DE MAYAGÜEZ

2013

Aprobada por:

Marko Schutz, Ph.D Miembro, Comité Graduado	Fecha
Omar Colón, Ph.D Miembro, Comité Graduado	Fecha
Dorothy Bollman, Ph.D Presidente, Comité Graduado	Fecha
Edusmildo Orozco, Ph.D Representante de Estudios Graduados	Fecha
Omar Colón, Ph.D Director del Departamento	Fecha

Resumen de Disertación Presentado a Escuela Graduada
de la Universidad de Puerto Rico como requisito parcial de los
Requerimientos para el grado de Maestría en Ciencias

SOBRE LA MULTIPLICACIÓN ESCALAR EN CURVAS ELÍPTICAS DEFINIDAS EN CUERPOS DE EXTENSIÓN ÓPTIMO

Por

Einstein Rafael Morales Morales

2013

Consejero: Dorothy Bollman Ph.D
Departamento: Ciencias Matemáticas

La seguridad de los sistemas de encriptación de clave pública está basada en la intratabilidad de ciertos problemas matemáticos, tales como la factorización de enteros que son producto de dos o más primos muy grandes. La encriptación de curva elíptica es un sistema de clave pública cuya seguridad está basada en la dificultad de encontrar el logaritmo discreto de un punto aleatorio en la curva elíptica con respecto al punto base público conocido. Una de las ventajas de la Criptografía de Curva Elíptica (CCE) es que esta puede ofrecer un alto nivel de seguridad con claves mucho más pequeñas que los sistemas tradicionales tales como RSA.

En muchas aplicaciones, los tiempos consumidos en la encriptación de datos pueden significar un gran problema. Por tanto, una tarea muy importante es optimizar el rendimiento de las operaciones en los sistemas de encriptación, así como una de las operaciones más importantes y de las que más tiempo consume en los sistemas de CCE que es la multiplicación escalar.

La mayoría de los esfuerzos para optimizar las operaciones de CCE hasta aquí, han sido dirigidas hacia la optimización de operaciones sobre cuerpos finitos de orden un

número primo o una potencia de 2.

Se han hecho muy pocos trabajos para operaciones sobre cuerpos finitos de orden una potencia de un primo mayor que 2. Una excepción es el trabajo de Daniel Bailey y Christof Paar quienes introdujeron el concepto de Cuerpos de Extensión Óptimo (CEO). Un CEO es un cuerpo $\mathbb{F}_{p^m}$ de orden p^m el cual puede ser definido por un binomio irreducible $x^m - \omega$ y donde p es un tipo especial de primo llamado un primo seudo Mersenne. Estos cuerpos son bien adecuados para implementaciones de "software" de sistemas CCE sobre máquinas de arquitectura fija.

En este trabajo consideramos sistemas CCE sobre una variación de CEOs que son más adecuadas para implementaciones de hardware e ilustramos nuestras ideas con una implementación en un FPGA. En vez de un primo seudo Mersenne requerimos primos de la forma $2^n - (2^i + 1)$, con tal primo p, la reducción módulo p puede ser llevada a cabo con un mínimo número de desplazamientos y sumas.

Dado un tamaño de "bit" s, consideramos el problema de determinar un primo p de la forma dada, un m tal que p^m es igual o cercano a s y un polinomio $f(x) = x^m - \omega$ que sea irreducible sobre $\mathbb{F}_p$ que optimice el rendimiento de las operaciones sobre el cuerpo $\mathbb{F}_{p^m}$. Un interés particular es elegir $m = 2$ ó $m = 3$ ya que en estos casos hay fórmulas simples para la reducción módulo $f(x)$ e inversos en $\mathbb{F}_{p^m}$.

Un cuerpo comunmente usado para CCE es $\mathbb{F}_{2^{163}}$, cuyos elementos tienen el tamaño de 163 "bits". Para un cuerpo de tamaño comparable, nosotros elegimos $p = 2^{54} - 33$ y $m = 3$. Los elementos de $\mathbb{F}_{(2^{54}-33)^3}$ tienen un tamaño de 162 "bits". En este caso elegimos el binomio irreducible $x^3 - 2$ de modo que las multiplicaciones en las reducciones módulo $f(x)$ puedan ser remplazadas por desplazamientos izquierdos.

Comparando nuestra implementación de la multiplicación escalar sobre un FPGA con otra implementación publicada sobre el cuerpo $\mathbb{F}_{2^{163}}$, observamos que nuestra implementación es en promedio 4 veces más rápida.

Abstract of Dissertation Presented to the Graduate School
of the University of Puerto Rico in Partial Fulfillment of the
Requirements for the Degree of Master of Sciences

ON POINT MULTIPLICATION IN ELLIPTIC CURVE OVER OPTIMAL EXTENSION FIELDS

By

Einstein Rafael Morales Morales

2013

Chair: Dorothy Bollman Ph.D
Major Department: Mathematical Sciences

The secuirty of public key encryption systems is based on the intractability of certain mathematical problems, such as factoring integers that are products of two or more very large primes. Elliptic curve cryptography (ECC) is a public key system whose security is based on the difficulty of finding the discrete logarithm of a random elliptic curve point with respect to a publicly known base point. One of the advantages of ECC is that it can yield a level of security with much smaller keys than traditional systems such as RSA.

In many applications, the time spent on data encryption can be a significant bottleneck. A very important task is thus to optimize the performance of encyrption system operations. The most important, as well as the most time consuming, operation in an ECC system is scalar multiplication. Most of the efforts at optimizing ECC operations so far have been directed toward optimizing operations over finite fields of order either a prime or a power of 2.

Very little work has been done for operations over finite fields of order a power of a prime. An exception is the work of Daniel Bailey and Christof Paar who introduced

the concept of optimal extension fields (OEFs). An OEF is a field $\mathbb{F}_{p^m}$ of order p^m which can be defined by an irreducible binomial $x^m - \omega$ and where p is a special type of prime called a pseudo-Mersenne prime. These fields are well suited for software implementations of ECC systems on machines of fixed word size.

In this work we consider ECC systems based on a variation of OEFs that is better suited for hardware implementations and we illustrate our ideas with an implementation in field programmable gate arrays (FPGAs). Instead of pseudo-Mersenne primes, we require primes of the form $2^n - (2^i + 1)$. With such a prime p reduction modulo p can be performed with a minimum number of shifts and sums. Given a bit size s we consider the problem of determining a prime of the given form, an m such that p^m is equal to or nearly equal to s and an irreducible binomial $f(x) = x^m - \omega$ over $\mathbb{F}_p$ that optimizes the performance of field operations over $\mathbb{F}_{p^m}$. Choosing $m = 2$ or $m = 3$ is of particular interest since in this case there are simple formulas for both reduction modulo $f(x)$ and $\mathbb{F}_{p^m}$ inverses.

One commonly used field for ECC is $\mathbb{F}_{2^{163}}$, whose elements have bit size 163. For a comparable size $\mathbb{F}_{p^m}$, we choose $p = 2^{54} - 33$ and $m = 3$. The elements of $\mathbb{F}_{p^m}$ have bit size 162. In this case, we can choose the irreducible binomial to be $f(x) = x^3 - 2$, so that the multiplications in reductions modulo $f(x)$ can be replaced by left shifts. Comparing our FPGA implementation to that of another published FPGA implementation of scalar multiplication over the field $\mathbb{F}_{2^{163}}$, we see that our implementation is on the average more than four times faster.

"*Por que lo escrito siempre perdura*"

A mis Padres...

A mis hermanos...

Y a ti mi amor.

AGRADECIMIENTOS

A Dios por llenarme de sabiduría y entendimiento.

Quiero dar un especial agradecimiento a la Dra. Dorothy Bollman por divulgarme sus conocimientos a lo largo de este camino y por brindarme la confianza de poder trabajar e investigar juntos en este trabajo.

A Roxana Barrios... estaré eternamente agradecido contigo, muchas gracias por todo de verdad.

A Edusmildo Orozco por su apoyo e interés constante en el presente trabajo.
Al departamento de Ciencias Matemáticas del Recinto Universitario de Mayagüez por darme esta gran oportunidad.
A los profesores del departamento de matemáticas por brindarme su conocimiento.
A David Marquez por sus oportunas sugerencias en la implementación.
A mis amigos Daiver, Gustavo, Lucho, Sindy, Carlos, Cesar, Robert, Fabián, Charlie y Shirley y demas compañeros de estudio que con sus chistes hicieron mas divertida esta carrera.
Al equipo de los viernes a las 5:00 pm frente a la India...
A las familias Morales Cabarcas, Morales Rodriguez y Barrios Rosales por sus mensajes de apoyo en la distancia.

Índice general

página

LISTA DE TABLAS

LISTA DE FIGURAS

LISTA DE ABREVIACIONES

AEB	Algoritmo Binario de Euclides
CE	Curva Elíptica
CCE	Criptografía de Curva Elíptica
CEO	Cuerpo de Extensión Óptimo
CEOFPGA	Cuerpo de Extensión Óptimo para FPGA
FFs	Flip-Flops
FPGA	Field Programable Gate Arrays
HDLs	Hardware Description Languages
ISE	Xilinx Integrated Software Environment
LSB	Least-Significant-Bit
LUTs	Look-up Tables
MSB	Most-Significant-Bit
NIST	National Institute of Standards and Technology
PLDCE	Problema del Logaritmo Discreto de Curva Elíptica
VHDL	Very-High-Speed Integrated Circuit Hardware Description Language

LISTA DE SÍMBOLOS

p	Número primo
$\mathbb{K}$	Cuerpo
$\mathbb{F}_p$	Cuerpo primo $\mathbb{F}$ de p elementos
$\sharp\,\mathbb{F}$	Cardinal del cuerpo $\mathbb{F}$
$\mathbb{F}_{2^m}$	Cuerpo binario de 2^m elementos
$\mathbb{F}_{p^m}$	Cuerpo con característica p
$\equiv$	Equivalente a
a^{-1}	Inverso de el número a
$gcd(a, b)$	Máximo comun divisor de los números a y b
$a(x)$	Polinomio de variable x
E	Curva Elíptica
Δ	Discriminante de una curva elíptica
∞	Punto infinito de una curva elíptica
$\mathbb{R}$	Números Reales
$E(\mathbb{K})$	Conjunto de puntos de una curva elíptica E sobre un cuerpo $\mathbb{K}$
$\sharp E(\mathbb{K})$	El número de elementos de una curva elíptica E definida sobre el cuerpo $\mathbb{K}$
P	Punto P sobre una curva elíptica E
kP	Multiplicación escalar en curva elíptica

Capítulo 1
CRIPTOGRAFÍA DE CLAVE PÚBLICA

En el siguiente capítulo, estudiaremos a nivel general conceptos de criptografía de clave pública y de como las curvas elípticas empezaron a jugar un papel importante en esta área. También, observaremos varias propiedades como la importancia que implica optimizar en los criptosistemas de curva elíptica, operaciones como la multiplicación escalar en una curva elíptica y lo útil que pueden ser los cuerpos de extensión óptimo y la tecnología de hardware reconfigurable al momento de hacer aritmética de cuerpo finito.

1.1. Criptosistemas asimétricos

Durante milenios, métodos como el de criptografía de clave privada y criptografía de clave pública, han ayudado a mantener ese deseo de tener siempre de forma secreta nuestra información, sin embargo, la aparición de este último, ha supuesto una auténtica revolución en la seguridad. En primer lugar, los algoritmos de clave pública están basados en funciones matemáticas, esto permite que el sistema de cifrado tenga propiedades especiales. En segundo lugar, este método criptográfico es asimétrico, es decir, utiliza dos claves de forma separada. En términos generales, todo sistema criptográfico consiste en dos algoritmos que dependen de una sola clave. Con una se cifra la información y con otra se descifra. Si utilizamos una misma clave para ambos algoritmos, estaremos hablando de criptografía de clave privada o símetrica. Si en cambio, usamos una clave para cifrar y otra distinta para descifrar,

entonces estaremos hablando de criptografía de clave pública o asimétrica.

El primer sistema criptográfico de clave pública que se dio a conocer, fue el conocido sistema de intercambio de claves de **Diffie-Hellman** [8], publicado en mayo de 1976. Este es un modelo abstracto de esquema criptográfico cuyo objetivo es el acuerdo de claves entre dos usuarios. El problema reside en averiguar la clave acordada por ambos y está basado en el Problema matemático del Logaritmo Discreto (PLD) [11].

Posteriormente, hacia 1978, apareció el primer sistema criptográfico llevado a la práctica, el RSA [1]. Su nombre se debe a sus creadores: **R**ivest, **S**hamir y **A**dleman. Este sistema se basa en el modelo de **Diffie-Hellman**, pero su fortaleza por este lado, está en tratar de resolver el problema de la factorización de enteros como base matemática .

Hoy en día este método persiste como uno de los sistemas más usado para cifrar, habiendo resistido multitud de ataques, sin embargo, debido a la aparición, en los últimos años, de algoritmos que resuelven este problema de la factorización en un tiempo menor al que se esperaba, se necesita agrandar el tamaño de las claves para hacer más fuerte este sistema. Pero ampliar el espacio de estas claves implicaría, usar sistemas de cómputos mucho más equipados, lo cual como se sabe, no están al alcance de todos los usuarios.

Como una opción, en 1985, **Neal Koblitz** [13] y **Victor Miller** [20] , independientemente, propusieron los Criptosistemas de Curva Elíptica (CCE), cuya seguridad descanza en el mismo problema que el método de Diffie-Hellman, pero en vez de usar números enteros como los símbolos del alfabeto del mensaje a encriptar, usa puntos de un objeto abstracto matemático llamado: ***Curva Elíptica***.

A partir de esta idea, se han propuesto y descubierto nuevas técnicas basadas en este mismo método, entre estas se resaltan algunas también muy conocidas como ElGamal y los esquemas de firma digital, como DSA.

La fascinación por las curvas elípticas en la seguridad informática y para los que las estudian, reside en la intratabilidad del problema del logaritmo discreto elíptico, en lo eficiente que pueden ser los CCE sobre ambientes restringidos en recursos como memoria, costo, velocidad etc. Y también, en la misma seguridad que brindan frente a métodos antiguos y tradicionales, como el RSA; ya que, por usar claves mucho más cortas, el costo en su empleo es mucho más reducido.

1.1.1. Criptografía de curva elíptica

Los criptosistemas ElGamal y el intercambio Diffie-Hellman podrían ser los esquemas más populares, cuando se trata de criptografía de curva elíptica, sin embargo antes de iniciar a detallar como funcionan estos métodos veamos que es una curva elíptica.

Una curva elíptica E viene definida por una ecuación de $Weierstrass$ de la forma:

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6.$$

donde los a_i pertenecen a un cuerpo $\mathbb{K}$, haciendo énfasis en los de mayor uso para fines criptográficos: los cuerpos finitos. (Ver capítulo 3).
Estos últimos son los que tendremos como objeto de estudio en este documento y en especial, se consideraran los Cuerpos de Extensión Óptimo que son cuerpos finitos $\mathbb{F}_q$ con $q = p^m$ elementos, donde p es un primo seudo Mersenne con $m \geqslant 1$ y están generados por un binomio $x^m - \omega$ irreducible sobre $\mathbb{F}_p$. Por otra parte, no se considera práctico usar curvas sobre cuerpos como el de los $\mathbb{R}$ debido a errores de redondeo y truncamiento de los valores [11].

Ahora, si nosotros queremos referirnos al conjunto de puntos de una curva elíptica E sobre un cuerpo $\mathbb{K}$, lo denotaremos por $E(\mathbb{K})$, que no será mas que el conjunto de puntos del producto cartesiano $\mathbb{K} \times \mathbb{K}$ que satisfagan la ecuacion de la curva, junto con el punto ∞, llamado infinito, el cual estaremos detallando mas adelante [4]. De forma más clara esto es:

$$E(\mathbb{K}) = \{(x, y) \in \mathbb{K} \times \mathbb{K} / y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6\} \cup \{\infty\}.$$

También tenemos, que si la característica del cuerpo $\mathbb{K}$ sobre el cual esté definida E es distinta de 2 y 3, nuestra curva podemos expresarla de la siguiente forma

$$y^2 = x^3 + ax + b \qquad a, b \in \mathbb{K}$$

con y si la característica solo es distinta de 2, usando transformaciones lineales adecuadas de las variables, se obtiene una de las siguientes expresiones [4]

$$y^2 + ay = x^3 + bx + c, \qquad a, b, c \in \mathbb{K}$$

$$y^2 + xy = x^3 + ax + b, \qquad a, b \in \mathbb{K}$$

Con estas nuevas expresiones veremos que el conjunto de soluciones, $E(\mathbb{K})$, de nuestra curva elíptica, tiene propiedades interesantes. En particular, se puede hacer del conjunto un *grupo*, es decir proveer al conjunto de una operación binaria y de un elemento identidad por medio de la ley de grupo para curvas elípticas, hecho que nos habilita para hacer criptografía.

A continuación se presenta el esquema de Diffie-Hellman sobre una Curva Elíptica.

1.1.2. Acuerdo de claves Diffie-Hellman

La criptografía de clave pública resuelve un problema importante que la criptografía de clave privada tiene como punto débil. Si A y B se quieren comunicar mediante un esquema simétrico los dos tienen que negociar una contraseña en común para cifrar y decifrar la información, generalmente esto se vuelve complejo si no existe un medio seguro por donde comunicar esta clave, ya que si existe un C que pueda filtrar esta clave de la comunicación, sería bastante sencillo para él descifrar el mensaje que A y B quisieran comunicar. El protocolo Diffie-Hellman es un criptosistema de clave pública que propone compartir una clave secreta entre dos partes, pero eligiendo cada parte una clave privada.

Si Alicia y Boris acuerdan utilizar una curva E definida sobre un cuerpo finito $\mathbb{K}$ y eligen un punto $P \in E(\mathbb{K})$. Entonces ellos pueden compartir la misma clave secreta como sigue.

Algoritmo 1 Protocolo de Diffie-Hellman en Curvas Elípticas

1: A escoge un entero positivo k_a que servirá como su clave privada.
2: B escoge un entero positivo k_b que servirá como su clave privada.
3: A calcula $P_a = k_a P$ y se lo envía a B.
4: B calcula $P_b = k_b P$ y se lo envía a A.
5: A calcula en secreto $Q = k_a P_b$
6: B calcula en secreto $R = k_b P_a$

Observe que A y B comparten las misma clave, ya que $Q = k_a(k_b P) = k_b(k_a P) = R$. Por tanto, ya ambos si lo desean pueden utilizar un esquema simétrico utilizando la misma clave. Note que solo necesitamos la multiplicación escalar para implementar el protocolo de Diffie-Hellman.

1.1.3. ElGamal de curva elíptica

El criptosistema ElGamal es un sistema de cifrado de clave pública, el cuál su seguridad está basada en la dificultad de calcular el logaritmo discreto. El **P**roblema del **L**ogaritmo **D**iscreto para una **C**urva **E**líptica (PLDCE) consiste en que: Dada

una curva elíptica E definida sobre un cuerpo finito $\mathbb{F}_q$, un punto $P \in E(\mathbb{F}_q)$ de orden n y un punto $Q \in < P >$, queremos encontrar si es posible un entero positivo s, tal que $Q = sP$ con $s < n$. El entero s es llamado el logaritmo discreto de Q para la base P y se denota por $s = \log_P Q$.

Por otra parte, el criptosistema ElGamal fue descrito por Taher Elgamal [9] en el año de 1984 y puede definirse sobre un grupo cíclico finito G. Este criptosistema consta de tres fases: generación de claves, proceso de cifrado y etapa de descifrado.

Algoritmo 2 Etapa 1: Generación de claves

1: Suponga que Alicia quiere enviar un mensaje a Boris. Para esto, Boris debe establecer una clave pública con la que le permitirá a Alicia enviar su mensaje. Inicialmente Boris escoge una curva elíptica E sobre un cuerpo finito $\mathbb{F}_{p^m}$, luego elige un punto $P \in E$ (usualmente se escoge de tal forma que el orden de P sea un primo bastante grande) y calcula $Q = sP$ donde s es un número entero secreto. Finalmente Boris publica la siguiente clave $[E, \mathbb{F}_{p^m}, P, Q]$.

Para que Alicia pueda enviar un mensaje a Boris debe hacer lo siguiente,

Algoritmo 3 Etapa 2: Cifrado

1: Tener la clave pública de Boris
2: Expresar su mensaje m como un punto $M \in E(\mathbb{F}_{p^m})$
3: Escoger un entero d y calcular $C_1 = dP$
4: Calcular $C_2 = M + dQ$
5: Enviar C_1, C_2 a Boris

Para que Boris obtenga el mensaje debe,

Algoritmo 4 Etapa 3: Descifrado

1: Calcular $C_2 - sC_1$

Esto funciona porque

$$C_2 - sC_1 = (M + dQ) - s(dP) = M + d(sP) - s(dP) = M.$$

Por otro lado, supongamos que una espía Eva desea infiltrar el sistema. Eva, sabe la información pública de Boris y los puntos C_1 y C_2. Si ella puede calcular logaritmos

discretos, puede usar P y Q para encontrar s y así descifrar el mensaje de la siguiente forma $C_2 - sC_1$. Ella también puede usar P y C_1 para encontrar d y calcular $M = C_2 - dQ$. Sin embargo, si no puede calcular logaritmos discretos, entonces no habra un camino para encontrar M. Además, es importante para Alicia usar un d aleatorio y distinto si desea enviar mensajes M y M' a Boris. Eva reconoce esto porque $C_1 = C_1'$. Así ella calcularía $C_2' - C_2 = M' - M$. Por ejemplo si M es un mensaje que se hace público un día después, entonces Eva averigua M, así ella calcula $M' = M - C_2 + C_2'$. Esto permite a Eva deducir el otro mensaje M' en este caso. [26]

Ejemplo 1.1.1 (ElGamal con curvas elípticas)**.**

Etapa 1

Boris elige la siguiente clave pública $[E, \mathbb{F}_p, P, Q]$, donde

$$\begin{aligned} E &: y^2 = x^2 + 102x + 2005 \\ p &= 314159265359 \\ P &= (228725321069,\ 127116812494) \\ Q &= (218896057517, 64059238278) \end{aligned}$$

Boris ha elegido su clave privada, escogiendo un entero secreto $s = 2718281828$ en el intervalo $(1, n)$ donde $n = 15707961439$ es el orden de P y computa $Q = s \cdot P = (218896057517, 64059238278)$.

Por otro lado, Alicia desea enviar a Boris el siguiente mensaje $m = 1234567890$.

Etapa 2

1. Alicia representa su mensaje $m = 1234567890$ como un punto

$$M = (1234567890, 259131096160) \in E(\mathbb{F}_p)$$

2. Luego escoge el entero $d = 2351458452$ en el intervalo $[1, 15707961438]$

3. y calcula los puntos

$$\begin{aligned} C_1 &= d \cdot P = (179839104564, 285023636671) \\ C_2 &= M + d \cdot Q = (182985347936, 293869714801) \end{aligned}$$

4. Finalemente, Alicia envía a Boris el par de puntos (C_1, C_2), es decir,

$$((179839104564, 285023636671), (182985347936, 293869714801))$$

Etapa 3

Boris descifra el mensaje

$$(C_1, C_2) = ((179839104564), 285023636671), (182985347936, 293869714801))$$

enviado por Alicia de la siguiente forma.

1. Boris calcula el punto $M = C_2 - s \cdot C_1$, es decir,

$$\begin{aligned} M &= (182985347936, 293869714801) - (299109926557, 212597623624) \\ &= (182985347936, 293869714801) + (299109926557, -212597623624) \\ &= (1234567890, 259131096160) \end{aligned}$$

2. Finalmente, Boris obtiene a partir de la abscisa del punto M el mensaje $m = 1234567890$ original.

1.2. Multiplicación escalar en CCE

No es difícil observar que en los criptosistemas de clave pública con curvas elipticas la operación kP donde k es un entero y P es un punto en la curva, es una de las operaciones más utilizadas. La multiplicación escalar en curvas elípticas es una de las operaciones que genera mayor interes entre los investigadores. Este interés reside en la posibilidad de diseñar algoritmos rápidos que permitan obtener el producto kP

en un tiempo razonable, ya que de esta depende que tan rápido es el criptosistema.

Por otro lado, existen algunos parámetros de dominio para curvas elípticas que permiten que el criptosistema sea fuerte a nivel de seguridad, estos parámetros dependen del cuerpo finito en el que esté definida la curva, por ejemplo existen estándares para hacer criptografía si los cuerpos son de la forma $\mathbb{F}_{2^m}$ y $\mathbb{F}_p$, es decir cuerpos binarios y cuerpos primos. Por ende, nuevas estrategias e implementaciones acerca de la multiplicación escalar en CEs han surgido con el fin de hacer más rápido los criptosistemas ya sea en implementaciones a nivel de hardware o software.

1.2.1. Aritmética de cuerpo finito en CCE

Es bien sabido que una eficiente aritmética de cuerpo finito es importante en una variedad de aplicaciones. Como se mencionó anterioremente en criptografía de curva elíptica existen estándares dependiendo en que cuerpo finito se encuentre definida la curva ya sea en un cuerpo binario $\mathbb{F}_{2^m}$ o un cuerpo primo $\mathbb{F}_p$. Dependiendo de las necesidades del usuario, él es libre de escoger que cuerpos y curvas usar.

A nivel de hardware trabajar con un cuerpo binario, podría resultar bastante útil a la hora de programar, esto por la naturaleza de la aritmética que también es binaria, sin embargo dado que los elementos del cuerpo son polinomios, las operaciones como la reducción módulo el polinomio irreducible que define el cuerpo $\mathbb{F}_{2^m}$, podrían resultar bastante costosas. Por otro lado, trabajar con cuerpos primos podría también resultar muy útil, ya que la aritmética es sobre números enteros y pues no existe el inconveniente de una reducción módulo un polinomio irreducible, sin embargo si los elementos del cuerpo son de gran tamaño, esto es en cantidad de "bits", la reducción módulo p que caracteriza el cuerpo primo y la multiplicación entre dos elementos puede resultar también costosas.

Actualmente, los Cuerpos de Extensión Óptimo (CEO) $\mathbb{F}_{p^m}$ están teniendo un gran interés, debido a las propiedades que estos presentan en su aritmética, los CEO son cuerpos de la forma $\mathbb{F}_{p^m}$ tal que p es un primo seudo Mersenne (esto es un primo de la forma $2^n \pm c$ con $\log_2 c \leq \lfloor \frac{n}{2} \rfloor$) y el polinomio que define el cuerpo es un binomio de la forma $x^m - \omega$. La aritmética en un CEO es eficiente por lo menos en software, debido a la eficiencia de la aritmética en el cuerpo $\mathbb{F}_p$ y a la simplicidad de la reducción módulo el binomio irreducible que define el cuerpo. Por este motivo los CEO y sus variaciones se convierten en una opción viable para trabajar CCE, ya que estos pueden proveer al sistema de propiedades que en cierto sentido ayuden a obtener mejores resultados de los que podrían brindar, los cuerpos binarios y primos, sobre todo si se trata de optimizar operaciones como la multiplicación escalar de curva elíptica.

En muchas aplicaciones, como por ejemplo las operaciones entre los elementos de una curva elíptica, se llevan a cabo usando operaciones aritméticas del cuerpo sobre el cual estén definidas. Aquí es donde la aritmética de cuerpo finito juega un papel esencial en el rendimiento de estas operaciones.

1.2.2. Field Programmable Gate Array (FPGA)

El hecho anterior, también nos indica que necesitamos de un ambiente adecuado en el cual implantar todas estas aplicaciones y operaciones. Un Field Programmable Gate Array (FPGA) es un circuito integrado que puede ser reconfigurado vía programación. Los FPGAs son programados usando un lenguaje de descripción de Hardware(HDL) tal como Verilog ó VHDL. Los FPGAs son usados cuando estos son requeridos para obtener un alto rendimiento en un costo rasonable usando algoritmos implementados en hardware, además cualquier circuito de aplicación específica

puede ser implementado en un FPGA, siempre y cuando esta disponga de los recursos necesarios.
Otras alternativas son circuitos de aplicación especificos (ASICs), sin embargo los FPGA's son más baratos y flexibles.

En general, a lo largo de este documento podremos observar algunas de las ventajas que implica implementar algoritmos sobre FPGA's tomando en cuenta las ventajas que presenta el usar nuestra propia versión de cuerpos de extensión óptimo. Así como también, una implementación de un multiplicador escalar en curvas elípticas definidas en cuerpos de extensión óptimo. Con el objetivo de verificar si los cálculos realizados para este trabajo fueron correctos, se validaron con el software matemático libre SAGEMATH [24] y la librería LiDIA que sirve para el desarrollo de teoría de números computacional y criptografía basada en C++. [14]

Capítulo 2
CUERPOS FINITOS

Una eficiente aritmética de cuerpo finito es de suma importancia para las aplicaciones de curvas elípticas, ya que las operaciones entre los elementos de la curva se llevan a cabo usando operaciones aritméticas del cuerpo sobre el cual la curva esté definida. En el siguiente capítulo hablaremos acerca de los cuerpos binarios, cuerpos primos y cuerpos de extensión óptimo, enfatizando nuestro interés en este último como cuerpos que en comparación con otros, pueden ayudar a agilizar procesos en operaciones de curva elíptica. También se describen algunas definiciones, teoremas y algoritmos acerca de la aritmética en cuerpos finitos que nos serán de utilidad en este estudio.

2.1. Conceptos generales

Los cuerpos son estructuras algebraicas tal que operaciones como la adición, sustracción, multiplicación y división están bien definidas. En matemática, los cuerpos resultan ser muy atractivos debido a que ayudan a la representación de elementos y entidades del mundo real.

Veamos a continuación formalmente lo que denotan estos objetos y como sus representaciónes nos pueden llegar a ser de utilidad.

Definición 2.1 (Cuerpo)**.** *Un cuerpo es un conjunto* $\mathbb{F}$ *que junto a dos operaciones, la adición y multiplicación denotadas por* $+$ *y* $\cdot$ *respectivamente, satisfacen las siguientes condiciones:*

1. $(\mathbb{F}, +)$ *es un grupo abeliano bajo la adición y tiene un elemento identidad denotado por* 0*.*
2. $(\mathbb{F}, 0, \cdot)$ *es un grupo abeliano bajo la multiplicación y tiene un elemento identidad denotado por* 1*.*
3. *Cumple la propiedad distributiva* $(a+b)\cdot c = a\cdot c + b\cdot c$ *para todo* $a, b, c \in \mathbb{F}$*.*

Si el conjunto $\mathbb{F}$ es finito, entonces el cuerpo se dice que es finito.

Como se mencionó anteriormente, en un cuerpo $\mathbb{F}$ también pueden ser definidas las operaciones de sustracción y división. La sustracción de elementos de $\mathbb{F}$ está definida en términos de la adición. Para a, $b \in \mathbb{F}$ se tiene que $a - b = a + (-b)$ donde $-b$ es el único elemento en $\mathbb{F}$ tal que $b + (-b) = 0$ y $-b$ es llamado el negativo de b. Similarmente la división de elementos en $\mathbb{F}$ es definida en términos de la multiplicación. Dados a, $b \in \mathbb{F}$ con $b \neq 0$ se tiene $a/b = a \cdot b^{-1}$ donde b^{-1} es el único elemento en $\mathbb{F}$ tal que $b \cdot b^{-1} = 1$ y b^{-1} es llamado el inverso de b [11].

Por otro lado, el número de elementos en un cuerpo finito se denomina *orden* de un cuerpo finito. Un cuerpo finito tendrá orden q si y solo si q es una potencia prima, esto es $q = p^m$ donde p es un número primo y m es un número entero positivo. Tal cuerpo lo denotaremos como $\mathbb{F}_{p^m}$ y p lo llamaremos la característica del cuerpo. En particular si $m = 1$, $\mathbb{F}_{p^m}$ se llama un cuerpo primo y si $p = 2$ entonces se llama un cuerpo binario.

Antes de pasar a describir un poco más los cuerpos finitos y su aritmética, es conveniente dar algunas definiciones previas de aritmética modular general.

Definición 2.2 (Congruencia)**.** *Dado* $1 < m \in \mathbb{Z}$*, se dice que* $a, b \in \mathbb{Z}$ *son congruentes módulo* m *si y solo si* $m|(a-b)$*. Esta relación se escribe como* $a \equiv b \ mod \ m$*, donde* m *es el módulo de la congruencia.*

La relación de congruencia módulo m es una relación de equivalencia para todo $m \in \mathbb{Z}$. Esto es, sean a, b, $c \in \mathbb{Z}$ entonces la relación de congruencia satisface las siguientes propiedades:

1. Reflexiva: $a \equiv a \bmod m$
2. Simétrica: $a \equiv b \bmod m$ entonces $b \equiv a \bmod m$
3. Transitiva: Si $a \equiv b \bmod m$ y $b \equiv c \bmod m$ entonces $a \equiv c \bmod m$

Usando división entera se puede probar que para enteros a, m con $m > 0$, existe un único par de enteros q y r tales que

$$a = mq + r$$

con $0 \leq r < m$. Se dice que q es el cociente y r es el residuo de dividir a por m. Es fácil observar que por definición $a \equiv r \bmod m$.

Es posible también sumar, restar, multiplicar, invertir y dividir módulo m para enteros a y b. El procedimiento generalmente se hace usando división entera para encontrar el residuo r y así usar la propiedad de que $a \equiv r \bmod m$. A continuación se muestran algunos ejemplos que ilustran cada una de estas operaciones:

1. Suma: $17 + 20 \equiv 15 \bmod 22$
2. Multiplicación: $2 \cdot 3 \equiv 2 \bmod 4$

Se dice que un entero a tiene un inverso módulo m si existe un entero b tal que $ab \equiv 1 \bmod m$. El entero b es el inverso de a y se escribe como a^{-1}. Usando la definición anterior, decimos que para dos enteros a y b la división de a por b se puede llevar a cabo calculando $a \cdot b^{-1} \bmod m$ donde b^{-1} es el inverso multiplicativo de b módulo m. Por ejemplo:

1. Inversión: El inverso de 2 módulo 5 es 3, ya que $2 \cdot 3 \equiv 1 \bmod 5$
2. División: $3/2 = 3 \cdot 2^{-1} \equiv 3 \cdot 3 \equiv 4 \bmod 5$

Por otro lado los elementos en el cuerpo $\mathbb{F}_{p^m}$ se comportan como polinomios de grado menor que m, por ejemplo si $a(x) = \sum_{i=0}^{m-1} a_i x^i$ y $b(x) = \sum_{i=0}^{m-1} b_i x^i$ son elementos de $\mathbb{F}_{p^m}$ entonces la suma $a(x)+b(x) = \sum_{i=0}^{m-1} c_i x^i$ donde $c_i = a_i + b_i \bmod p$ está bien definida porque $a(x) + b(x)$ es un polinomio de grado menor que m y por tanto pertenece a $\mathbb{F}_{p^m}$. Para el producto el procedimiento es similar, solo que además de que los coeficientes se reducen módulo p, también se necesita hacer una reducción módulo un polinomio irreducible sobre el cuerpo $\mathbb{F}_p$.

A continuación en las siguientes secciónes se describen los cuerpos finitos binarios, primos y otros muy especiales como lo son los cuerpos de extensión óptimo.

2.2. Cuerpos binarios $\mathbb{F}_{2^m}$

El cuerpo denotado por $\mathbb{F}_{2^m}$ es conocido como el cuerpo finito binario o de característica dos.

Un cuerpo finito binario $\mathbb{F}_{2^m}$ es una extensión del cuerpo base $\mathbb{F}_2 = \{0,\ 1\}$. Los elementos que pertenecen a este cuerpo se pueden representar como polinomios

de grado a lo más $m-1$ y tienen como coeficientes valores en $\{0,\ 1\}$ es decir,

$$a(x) = a_{m-1}x^{m-1} + \cdots + a_1 x + a_0 = \sum_{i=0}^{m-1} a_i x^i \ \text{ donde } \ a_i \in \{0,\ 1\}$$

Las operaciones en un cuerpo finito binario son operaciones modulares bajo un polinomio irreducible llamado también polinomio generador, este polinomio es utilizado para generar dicho cuerpo y tiene la forma:

$$f(x) = x^m + \sum_{i=1}^{m-1} a_i x^i + 1 \ \text{ donde } \ a_i \in \{0,\ 1\}$$

Note que este polinomio es de grado m y al ser irreducible significa que no puede ser factorizado como un producto de polinomios en $\mathbb{F}_{2^m}$.

El siguiente ejemplo describe los elementos de $\mathbb{F}_{2^3}$ y las operaciones se hacen módulo el polinomio irreducible $f(x) = x^3 + x + 1$.

El cuerpo $\mathbb{F}_{2^3}$ tiene como elementos

$$\begin{aligned} \mathbb{F}_{2^3} &= \{a_2\alpha^2 + a_1\alpha + a_0 \mid a_2,\ a_1,\ a_0 \in \mathbb{F}_2 \text{ y } f(\alpha) = 0\} \\ &= \{0, 1, \alpha, \alpha^2, \alpha+1, \alpha^2+1, \alpha^2+\alpha+1, \alpha^2+\alpha\} \end{aligned}$$

donde $\alpha^2 + \alpha + 1 = 0$

Por conveniencia, el polinomio $a_2\alpha^2 + a_1\alpha + a_0 \in \mathbb{F}_{2^3}$ es representado por el vector (a_2, a_1, a_0) de longitud 3, lo cual resulta ser adecuado para una representación en hardware.

Las operaciones de suma, multiplicación y división están bien definidas sobre los cuerpos binarios. En el siguiente ejemplo se muestran tres tablas en las que se indican cuales son las posibles sumas, multiplicaciones e inversiones sobre el cuerpo

$\mathbb{F}_{2^2}$ definido por el polinomio irreducibible $f(x) = x^2 + x + 1$.

Ejemplo 2.2.1.

$+$	00	01	10	11
00	00	01	10	11
01	01	00	11	10
10	10	11	00	01
11	11	10	01	00

Tabla 2–1: Suma en $\mathbb{F}_{2^2}$

$*$	00	01	10	11
00	00	00	00	00
01	00	01	10	11
10	00	10	11	01
11	00	11	01	10

Tabla 2–2: Multiplicación en $\mathbb{F}_{2^2}$

a	a^{-1}
01	01
10	11
11	10

Tabla 2–3: Inversión en $\mathbb{F}_{2^2}$

Para los cuerpos binarios existen algoritmos bastante eficientes que modelan cada una de estas operaciones. Por ejemplo, para la multiplicación de dos elementos $a(\alpha)$ y $b(\alpha)$ en $\mathbb{F}_{2^m}$, tenemos algoritmos como la multiplicación de Karasuba-Ofman, el MSB y LSB multiplier del inglés Most-Significant-Bit y Least-Significant-Bit, multiplicación de Mastrovito, multiplicación de Montgomery etc. [7]. Todos muy populares y eficientes en esta área. Así mismo, también existen algoritmos para hallar inversos como el algoritmo extendido de Euclides, el algoritmo binario, el de Itoh-Tsujii etc. [7].

Los cuerpos binarios también resultan ser muy interesantes por su naturaleza

binaria, ya que son muy eficientes a la hora de programar aritmética de hardware. Sin embargo, el costo en la operación reducción módulo el polinomio irreducible podría resultar algo costosa si el exponente m en $\mathbb{F}_{2^m}$ es grande. Estas desventajas hace que nuevos cuerpos sean estudiados, cuerpos en los que no haya necesidad de hacer reducción módulo un polinomio irreducible, ejemplo de esto son los cuerpos primos que también resultan ser muy interesantes a la hora de hacer aritmética de cuerpo finito. Veamos entonces a continuación lo que denotan estos cuerpos.

2.3. Cuerpos primos $\mathbb{F}_p$

Inicialmente se mencionó que un cuerpo finito $\mathbb{F}_{p^m}$ es primo si se cumple que $m = 1$.

Definición 2.3 (Cuerpo primo)**.** *Los enteros módulo un primo p, es decir el conjunto de enteros* $\{0, 1, 2, \cdots, p-1\}$ *donde la adición y multiplicación entre ellos se realiza módulo p, es un cuerpo finito de orden p, el cual denotaremos como el cuerpo* $\mathbb{F}_p$ *y se denomina cuerpo primo.*

La aritmética sobre un cuerpo $\mathbb{F}_p$ es aritmética módulo p y las operaciones de suma, multiplicación y división están bien definidas. En el siguiente ejemplo se muestran tres tablas en las que se indican cuales son las posibles sumas, multiplicaciones e inversiones sobre el cuerpo $\mathbb{F}_7$

A continuación veremos algunas definiciones y algoritmos que nos serviran para poder desarrollar aritmética en cuerpos $\mathbb{F}_p$

+	0	1	2	3	4	5	6
0	0	1	2	3	4	5	6
1	1	2	3	4	5	6	0
2	2	3	4	5	6	0	1
3	3	4	5	6	0	1	2
4	4	5	6	0	1	2	3
5	5	6	0	1	2	3	4
6	6	0	1	2	3	4	5

Tabla 2–4: Adición en $\mathbb{F}_7$

$*$	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6
2	0	2	4	6	1	3	5
3	0	3	6	2	5	1	4
4	0	4	1	5	2	6	3
5	0	5	3	1	6	4	2
6	0	6	5	4	3	2	1

Tabla 2–5: Multiplicación en $\mathbb{F}_7$

a	a^{-1}
1	1
2	4
3	5
4	2
5	3
6	6

Tabla 2–6: Inversión en $\mathbb{F}_7$

2.3.1. Aritmética módulo p

Dado que un elemento a en un cuerpo finito $\mathbb{F}_p$ se puede considerar como un entero a tal que $0 \leq a < p$, se pueden desprender a partir de este hecho varias propiedades que nos facilitarán un poco más la aritmética sobre estos cuerpos.
A continuación describiremos algunas de estas operaciones.

2.3.1.1. Adición y Sustracción

La implementación de las operaciones de adición y sustracción de dos elementos en $\mathbb{F}_p$ pueden resultar bastante sencillas mediante una adición o una sustracción y

luego realizar la reducción modular substrayendo p o adicionando p solo una vez de el resultado intermedio.

Si tomamos dos elementos $x,\ y$ en $\mathbb{F}_p$ entonces $z = (x + y) \bmod p$, será igual a los valores $x + y$ o $x + y - p$.

Note que esto es válido por que $0 \leq x + y < 2p$. El siguiente algoritmo a continuación calcula el valor de z.

Algoritmo 5 Adición módulo p: Forma clásica.

Entrada: $x,\ y \in \mathbb{F}_p$
Salida: $z = x + y \bmod p$

1: $z_1 \leftarrow x + y$
2: $z_2 \leftarrow z_1 - p$
3: **si** $z_2 \geqslant 0$ **entonces**
4: $\quad z \leftarrow z_2$
5: **si no**
6: $\quad z \leftarrow z_1$
7: **fin si**

El método anterior es la forma clásica de sumar dos elementos en un cuerpo finito $\mathbb{F}_p$ el cual resulta ser bastante sencillo, sin embargo en [7] se menciona que el algoritmo anterior puede ser modificado ligeramente usando en el proceso, divisiones con potencias de 2, de tal forma que se pueda obtener un circuito más eficiente a nivel de hardware.

Supongamos que p es un número de k "bits" y definamos $z_2 = (z_1 \bmod 2^k) + (2^k - p)$, en vez de $z_1 - p$ como se había definido previamente.

Entonces, consideremos los siguientes 3 casos:

1. Si $z_1 < p$, de modo que $z_1 < 2^k$, entonces $z_2 = z_1 + (2^k - p) < p + (2^k - p) = 2^k$

2. Si $p \leq z_1 < 2^k$ entonces

$$z_2 = z_1 + (2^k - p) \geq p + (2^k - p) = 2^k \text{ y } z_2 \mod 2^k = z_1 - p$$

3. Si $2^k \leq z_1$, de modo que $p < z_1$, entonces

$$z_2 = (z_1 - 2^k) + (2^k - p) = z_1 - p < 2p - p = p < 2^k \text{ y } z_2 \mod 2^k = z_1 - p$$

En resumen,

Si $z_1 < p$, entonces $z_1 < 2^k$ y $z_2 < 2^k$

Si $p \leq z$, entonces o $z_1 \geq 2^k$ o $z_2 \geq 2^k$ y $z_1 - p = z_2 \bmod 2^k$

Lo anterior se resume en el siguiente algoritmo

Algoritmo 6 Adición módulo p: Forma binaria

Entrada: $x, y \in \mathbb{F}_p$
Salida: $z = x + y \bmod p$
1: $z_1 \leftarrow x + y$; $z_2 \leftarrow (z_1 \mod 2^k) + (2^k - p)$
2: $c_1 \leftarrow z_1/2^k$; $c_2 \leftarrow z_2/2^k$
3: **si** $c_1 = 0$ y $c_2 = 0$ **entonces**
4: $\quad z \leftarrow (z_1 \mod 2^k)$
5: **si no**
6: $\quad z \leftarrow (z_2 \mod 2^k)$
7: **fin si**

Por otro lado, para la sustracción de elementos en el cuerpo $\mathbb{F}_p$ el procedimiento es algo similar a el de la adición.

Si tomamos elementos x, y en $\mathbb{F}_p$ entonces $z = (x - y) \bmod p$ será igual a los valores $x - y$ o $x - y + p$.

Note que esto es válido por que $-p < x - y < p$. El siguiente algoritmo a continuación calcula el valor de z.

Algoritmo 7 Sustracción módulo p: Forma clásica.

Entrada: $x, y \in \mathbb{F}_p$
Salida: $z = x - y \bmod p$
1: $z_1 \leftarrow x - y$; $z_2 \leftarrow z_1 + p$
2: **si** $z_1 < 0$ **entonces**
3: $\quad z \leftarrow z_2$
4: **si no**
5: $\quad z \leftarrow z_1$
6: **fin si**

Al igual que con el caso de la suma, en [7] también se indica que el algoritmo anterior puede ser modificado usando en el proceso, divisiones con potencias de 2, con el objetivo que se pueda obtener un circuito más eficiente a nivel de hardware. Supongamos que $z_1 = (x - y + 2^k) \bmod 2^k$ en vez de $x - y$, como se había definido previamente. Entonces consideremos dos casos:

$$Si\ -p < x - y < 0,\ entonces\ z_1 = x - y + 2^k < 2^k,$$

$$z_2 = z_1 + p = x - y + 2^k + p > 2^k$$

$$z_2 \mod 2^k = x - y + p$$

$$Si\ 0 \leq x - y < p,\ entonces\ 2^k \leq x - y + 2^k < 2^k + p,\ z_1 = x - y$$

Es decir, $x - y + 2^k < 2^k$ y $z = z_2 \mod 2^k$, ó $2^k \leq x - y + 2^k$ y $z = z_1$. Lo anterior se resume en el siguiente algoritmo.

Algoritmo 8 Sustracción módulo p: Forma binaria.

Entrada: $x, y \in \mathbb{F}_p$
Salida: $z = x - y \bmod p$
1: $s \leftarrow x + (2^k - y)$
2: $z_1 \leftarrow (s \mod 2^k)$; $c_1 \leftarrow s/2^k$
3: $z_2 \leftarrow (z_1 + p) \mod 2^k$
4: **si** $c_1 = 1$ **entonces**
5: $\quad z \leftarrow z_1$
6: **si no**
7: $\quad z \leftarrow z_2$
8: **fin si**

Por otro lado, en la siguiente sección se describe la multiplicación de dos elementos en $\mathbb{F}_p$.

2.3.1.2. Multiplicación módulo p

El algoritmo que se discute en esta sección propuesto en [22], es la versión clásica para obtener el producto de dos enteros a y b mediante sumas repetitivas que dependen de la representación binaria de cada uno de ellos. Note que el producto se obtiene sumando una traslación del multiplicando para cada "bit" no cero del multiplicador. Observe también que para este producto aún no se ha determinado la reducción modular.

Algoritmo 9 Multiplicación ordinaria

Entrada: a y b enteros de k y l bits (resp.)
Salida: $a \cdot b$

1: /* $x[i]$ representa el i-ésimo bit de x */
2: max$\leftarrow k + l - 1$
3: $prod \leftarrow 0$
4: **para** $i = 0$ hasta max **hacer**
5: **si** $i < k$ **entonces** $x[i] \leftarrow a[i]$ **si no** $x[i] \leftarrow 0$ **fin si**
6: **fin para**
7: **para** $i = 0$ hasta $l - 1$ **hacer**
8: **si** $b[i]$=1 **entonces** $prod \leftarrow prod + x$; **fin si**
9: $x \leftarrow x \ll 1$ /* "$\ll$" traslada una posición hacia la izquierda */
10: **fin para**
11: **devolver** $prod$

Existen muchos otros algoritmos que nos permiten determinar el producto de dos elementos en un cuerpo primo. En [7, 23] proponen algoritmos como Karatsuba-Ofman, Mastrovito, Montgomery etc. donde se presentan resultados acerca del costo y retraso que implica cada uno de estos algoritmos.

El producto anterior de los dos elementos a y b en $\mathbb{F}_p$ podría ser costoso si el tamaño en "bits" es grande. Sin embargo falta algo importante en este producto y es la reducción módulo p. Podríamos pensar en el método de la división larga, sin

embargo este es el método menos recomendable para una reducción modular por lo costoso que puede ser a nivel de hardware.

La siguiente sección intenta explicar cual sería la forma más apropiada para hacer una reducción módulo p.

2.3.1.3. Reducción módulo p

Para números primos p que no tienen una forma especial, la reducción de un elemento x mod p puede convertirse en un procedimiento laborioso y costoso.

En criptografía de curva elíptica, el rendimiento de los criptosistemas dependen estrechamente de la rapidez de la multiplicación del cuerpo y de una singular y eficiente forma de reducir elementos en el cuerpo.

Como se mencionó anteriormente, es claro que una primera estrategia de reducción es la división larga y luego tomar el residuo, sin embargo este también se convierte en uno de los primeros métodos menos adecuados para el diseño de aplicaciones criptográficas.

Como se mencionó anteriormente, una singular y especial escogencia de un primo p permitiría que la reducción modular sea menos compleja.

Es bien sabido, que hay enteros que permiten una rápida o menos compleja reducción modular sin divisiones. Esto es posible, cuando la reducción la queremos hacer módulo un entero que tiene la forma $2^n \pm c$ donde c es un valor entero positivo relativamente pequeño. Por tanto, si nuestro valor de p es un primo de la forma $2^n - c$, entonces la reducción en el cuerpo la podemos hacer de la siguiente manera.

En [4] se propone el siguiente algoritmo que permite reducir un entero $x < p^2$, ideal para productos de dos elementos que estén en un cuerpo $\mathbb{F}_p$.

Algoritmo 10 Reducción Modular $p = 2^n - c$

Entrada: $p = 2^n - c$, $\log_2 c \leq \lfloor \frac{1}{2} n \rfloor$, $0 \leq x < p^2$.
Salida: $r \equiv x \bmod p$

1: $q_0 \leftarrow x >> n$
2: $r_0 \leftarrow x - q_0 2^n$
3: $r \leftarrow r_0$
4: $i \leftarrow 0$
5: **mientras** $q_i > 0$ **hacer**
6: $\quad q_{i+1} \leftarrow q_i c >> n$
7: $\quad r_{i+1} \leftarrow q_i c - (q_{i+1} >> n)$
8: $\quad i \leftarrow i + 1$
9: $\quad r \leftarrow r + r_i$
10: **fin mientras**
11: **mientras** $r \geq p$ **hacer**
12: $\quad r \leftarrow r - p$
13: **fin mientras**

El algoritmo que se da en el siguiente teorema está basado en la idea del algoritmo 10, pero es más apropiado para la implementación en hardware. Sin embargo, podría ser costoso, a menos que c tenga una forma especial (que consideraremos más adelante). Es importante notar que si x es un número consistiendo de $2n$ "bits", entonces los primeros n "bits", $x[1]$, representan el cociente cuando x se divide entre 2^n y los otros n bits, $x[0]$, representan el residuo.

Teorema 2.1. *Sea $p = 2^n - c$, donde $\log_2 c \leq \lfloor \frac{1}{2} n \rfloor$ un primo seudo Mersenne y sea $x < p^2$. Si $n > 4$, entonces x se puede reducir módulo p como sigue:*

1. $x' \leftarrow x[1] \cdot c + x[0]$
2. $x'' \leftarrow x'[1] \cdot c + x'[0]$
3. *Si* $x'' > p$ *entonces* $x'' \leftarrow x'' - p$
4. *Si* $x'' > p$ *entonces* $x'' \leftarrow x'' - p$

donde $x[0]$ y $x[1]$ representan bloques de 2^n "bits".

Prueba: Supongamos que $x < p^2 = 2^{2n} - 2c2^n + c^2$. Como $2^n > c$ y $c > 0$ entonces $c2^n > c^2$, luego $2c2^n > c^2$ osea que $0 > -2c2^n + c^2$, esto nos dice que $2^{2n} - 2c2^n + c^2 < 2^{2n}$, osea que

$$x < 2^{2n} \tag{2.1}$$

Ahora por el algoritmo de la división $x = x[1] \cdot 2^n + x[0]$ donde $x[0], x[1] < 2^n$ (ya que si $x[1] > 2^n$ se contradice 2.1), luego $x \equiv x[1] \cdot c + x[0]$ *mod* p (*paso 1*), osea que $x < 2^n(c+1)$.

Usando nuevamente el algoritmo de la división, tenemos que $x[1] \cdot c + x[0] = x'[1] \cdot 2^n + x'[0]$ donde $x'[1] \leq c$ y $x'[0] < 2^n$, luego $x[1] \cdot c + x[0] \equiv x'[1] \cdot c + x'[0]$ *mod* p (*paso 2*), osea que $x[1] \cdot c + x[0] \leq c^2 + 2^n \leq 2^{n+1}$. Pero como $c < 2^{\frac{n}{2}}$, se tiene que $3c < 3 \cdot 2^{\frac{n}{2}} < 2^{\frac{n}{2}+2} < 2^n$ para $n > 4$, osea que $2^n - 3c > 0$, por tanto $x'[1] \cdot c + x'[0] < 2^{n+1} + 2^n - 3c = 3p$. Es decir que después de a lo sumo 2 sustracciones (*paso 3 y 4*), el valor de x estará reducido módulo p.

Por otro lado, en [11, 21] se recomiendan algunos primos que por su forma particular permiten producir algoritmos de rápida reducción modular.

Los estándares para desarrollar criptografía de curva elíptica, recomiendan que las curvas sean definidas sobre los siguientes cuerpos primos donde la característica de cuerpo es:

$$\begin{aligned}
p_{192} &= 2^{192} - 2^{64} - 1 \\
p_{224} &= 2^{224} - 2^{96} + 1 \\
p_{256} &= 2^{256} - 2^{224} + 2^{192} + 2^{96} - 1 \\
p_{384} &= 2^{384} - 2^{128} - 2^{96} + 2^{32} - 1 \\
p_{521} &= 2^{521} - 1
\end{aligned}$$

Como un ejemplo de un método de reducción para estos primos especiales, presentamos el caso p_{192} .

Ejemplo 2.3.1 (Reducción módulo $p = 2^{192} - 2^{64} - 1$)**.** *Sean* $p = 2^{192} - 2^{64} - 1$ *y* x *un entero tal que* $0 \leq x < p^2$*. Sea*

$$x = x_5 2^{320} + x_4 2^{256} + x_3 2^{192} + x_2 2^{128} + x_1 2^{64} + x_0$$

que no es más que la representación de x *en base* 2^{64}*, donde cada* $0 \leq x_i < 2^{64}$*. Ahora utilizando las siguientes congruencias*

$$\begin{aligned} 2^{192} &\equiv 2^{64} + 1 \\ 2^{256} &\equiv 2^{128} + 2^{64} \\ 2^{320} &\equiv 2^{128} + 2^{64} + 1 \end{aligned}$$

tenemos que

$$x' \equiv x_5 2^{128} + x_5 2^{64} + x_5 + x_4 2^{128} + x_4 2^{64} + x_3 2^{64} + x_3 + x_2 2^{128} + x_1 2^{64} + x_0$$

Una cota superior para x' *está dado por* $x' < 2^{192} + (2^{192} - 2^{64}) + 2^{128} + 2^{192} < 4(2^{192} - 2^{64} - 1)$*. Así que,* x *mod* p *es*

$$x', \ \acute{o}\ x' - p, \ \acute{o}\ x' - 2p, \ \acute{o}\ x' - 3p$$

En [11] se propone el siguiente algoritmo para la reducción módulo p_{192}

Algoritmo 11 Reducción modular $p = 2^{192} - 2^{64} - 1$

Entrada: $x = (x_5, x_4, x_3, x_2, x_1, x_0)$ en base 2^{64} con $x < p^2$.
Salida: $x \mod p$

1: Definir los enteros:
2: $s_1 \leftarrow (x_2, x_1, x_0), s_2 \leftarrow (0, x_3, x_3)$
3: $s_3 \leftarrow (x_4, x_4, 0), s_4 \leftarrow (x_5, x_5, x_5)$
4: **devolver** $s_1 + s_2 + s_3 + s_4 \mod p$

Analogamente, se trabaja para los demás primos sugeridos.

Resultaría interesante encontrar propiedades más generales para una reducción modular, en la que el primo p sea de la forma $2^n - 2^i - 1$ con $i < n$. Es posible que se tengan que hacer más sumas adicionales, en comparación con casos ideales como los que acabamos de ver, sin embargo los primos p_{192}, p_{224}, p_{256}, p_{384}, p_{521} ya han sido estudiados en la mayoría de los casos de CCE sobre cuerpos primos, por ende resultaría víable proponer una nueva familia de cuerpos que permitan variar este estudio ya conocido.

2.3.1.4. División entera módulo p

El inverso de un elemento $a \in \mathbb{F}_p$ distinto de cero, denotado por $a^{-1} \bmod p$ o simplemente a^{-1}, es el único elemento $x \in \mathbb{F}_p$ tal que $ax = 1$ en $\mathbb{F}_p$ esto es $ax \equiv 1 \bmod p$. Los inversos pueden ser calculados eficientementes usando el algoritmo extendido de Euclides para números enteros, sin embargo también existen algoritmos muy eficientes como el algoritmo binario que permite encontrar inversos módulo p.

Dados dos enteros a y b, el algoritmo de Euclides puede ser extendido para encontrar enteros x y y tales que $ax + by = d$ donde $d = \gcd(a, b)$.

El siguiente algoritmo mantiene las invariantes $ax_1 + by_1 = u$, $ax_2 + by_2 = v$ siempre que $u \leqslant v$. El algoritmo termina cuando $u = 0$, en cuyo caso $v = \gcd(a, b)$ y $x = x_2$, $y = y_2$ satisfacen $ax + by = d$.

Algoritmo 12 Algoritmo extendido de Euclides para enteros

Entrada: Enteros positivos a y b con $a \leq b$;
Salida: $d = \gcd(a, b)$ y enteros x, y que satisfacen $ax + by = d$.

1: $u \leftarrow a$, $v \leftarrow b$
2: $x_1 \leftarrow 1$, $y_1 \leftarrow 0$, $y_2 \leftarrow 1$.
3: **mientras** $u \neq 0$ **hacer**
4: $\quad q \leftarrow \lfloor v/u \rfloor$, $r \leftarrow v - qu$, $x \leftarrow x_2 - qx_1$, $y \leftarrow y_2 - qy_1$.
5: $\quad v \leftarrow u$, $u \leftarrow r$, $x_2 \leftarrow x_1$, $x_1 \leftarrow x$, $y_2 \leftarrow y_1$, $y_1 \leftarrow y$.
6: **fin mientras**
7: $d \leftarrow v$, $x \leftarrow x_2$, $y \leftarrow y_2$
8: **devolver** (d, x, y)

El algoritmo extendido de Euclídes es uno de los algoritmos más clásicos, para hallar inversos sobre $\mathbb{F}_p$. Si tomamos ahora un primo p y un entero a tal que $a \in [1, p-1]$, dado que $\gcd(a, p) = 1$, si ejecutaramos el algoritmo de Euclides con las entradas a y p, cuando los enteros $u,\ x_1$ y y_1 satisfagan $ax_1 + py_1 = u$ con $u = 1$. Se tendrá entonces que $ax_1 \equiv 1 \bmod p$ y por tanto $a^{-1} \equiv x_1 \bmod p$.

Algoritmo 13 Inversión en $\mathbb{F}_p$ usando el Algoritmo Extendido de Euclides

Entrada: Primo p y $a \in [1, p-1]$
Salida: $a^{-1} \bmod p$
1: $u \to a$, $v \to p$
2: $x_1 \to 1$, $x_2 \to 0$
3: **mientras** $u \neq 1$ **hacer**
4: $\quad q \leftarrow \lfloor v/u \rfloor$, $r \leftarrow v - qu$, $x \leftarrow x_2 - qx_1$.
5: $\quad v \leftarrow u$, $u \leftarrow r$, $x_2 \leftarrow x_1$, $x_1 \leftarrow x$
6: **fin mientras**
7: **devolver** $(x_1 \bmod p)$

Veamos un ejemplo para encontrar el inverso de un número usando el algoritmo extendido de Euclides

Ejemplo 2.3.2. *Sea $p = 11$ y $a = 8$, usando el algorítmo extendido de Euclides encontremos el inverso multiplicativo de $8 \, mod \, 11$, vemos que*

$$\begin{aligned} 11 &= 8(1) + 3 \to 3 = 11 - 8(1) \\ 8 &= 3(2) + 2 \to 2 = 8 - 3(2) \\ 3 &= 2(1) + 1 \to 1 = 3 - 2(1) \\ 2 &= 1(2) \end{aligned}$$

remplazando las anteriores ecuaciones se tiene que,

$$\begin{aligned} 1 &= 3 - 2(1) \\ 1 &= 3 - (8 - 3(2))(1) = 3 - (8 - 3(2)) = 3(3) - 8 \\ 1 &= (11 - 8(1))(3) - 8 = 11(3) + 8(-4) \end{aligned}$$

Luego podemos ver que $1 \equiv 8(-4)$ *mod* 11 *o sea que* $1 \equiv 8(7)$ *mod* 11.

Sin embargo, el algoritmo Extendido de Euclides es costoso computacionalmente por su requerimientos de división.

El algoritmo binario resuelve estas divisiones y las reemplaza por desplazamientos, ya que este opera con sustracciones y divisiones por 2.

A continuación se muestra el algoritmo binario de división entera.

Algoritmo 14 Algoritmo binario para la inversión en $\mathbb{F}_p$

Entrada: Primo p y $a \in [1, p-1]$.
Salida: $a^{-1} \bmod p$

1: $u \leftarrow a$, $v \leftarrow p$
2: $x_1 \leftarrow 1$, $x_2 \leftarrow 0$.
3: **mientras** $u \neq 1$ y $v \neq 1$ **hacer**
4: **mientras** u es par **hacer**
5: $u \leftarrow u/2$
6: **Si** x_1 es par **entonces** $x_1 \leftarrow x_1/2$; **sino** $x_1 \leftarrow (x_1 + p)/2$.
7: **fin mientras**
8: **mientras** v es par **hacer**
9: $v \leftarrow v/2$
10: **Si** x_2 es par **entonces** $x_2 \leftarrow x_2/2$; **sino** $x_2 \leftarrow (x_2 + p)/2$.
11: **fin mientras**
12: **Si** $u \geq v$ **entonces** $u \leftarrow u - v$, $x_1 \leftarrow x_1 - x_2$;
13: **Sino** $v \leftarrow v - u$, $x_2 \leftarrow x_2 - x_1$.
14: **fin mientras**
15: **Si** $u = 1$ **entonces devolver** $(x_1 \bmod p)$; **sino devolver** $(x_2 \bmod p)$

Veamos ahora un ejemplo para encontrar el inverso de un número usando el algoritmo binario

Ejemplo 2.3.3. *Sea* $p = 11$ *y* $a = 8$,*encuentre el inverso multiplicativo de* 8 *mod* 11 *por medio del algoritmo binario*

u	v	x_1	x_2
8	11	1	
4		6	
2		3	
1		7_*	
	10		-7

Tabla 2–7: Inverso de 8 mod 11

luego como $u = 1$, entonces el inverso de 8 módulo 11 es 7.

La sección que acabamos de ver nos da una idea de lo interesante que es estudiar aritmética modular. Veamos a continuación otra clase de cuerpos finitos que resultan ser interesantes, ya que algunas de sus operaciones también se basan en la arimética módulo p. Pasemos a describir los cuerpos de extensión óptimo.

2.4. Cuerpos de Extensión Óptimo $\mathbb{F}_{p^m}$

Las secciones anteriores nos muestran que no cabe duda que la aritmética de cuerpo finito merece un estudio muy profundo y que es un punto primordial en el estudio de las implementaciones de hardware. Se observó que los cuerpos binarios pueden resultar bastante atractivos para este tipo de implementaciones ya que sus operaciones pueden involucrar desplazamientos y aritmética binaria de hardware. Sin embargo, las operaciones como la multiplicación y la reducción módulo el polinomio irreducible que genera el cuerpo, pueden resultar un poco lentas así como sucede con algunas de las operaciones en los cuerpos primos que debido al tamaño

de la característica la reducción módular puede ser altamente dispendiosa.

Los Cuerpos de Extensión Óptimo tratan de disminuir las desventajas que cada uno de los cuerpos anteriores presenta y se colocan dentro de los cuerpos finitos como una nueva opción viable para hacer aritmética eficiente.
Ventajas como hacer reducción módulo un binomio irreducible y reducción entera módulo un primo con ciertas características especiales, son los que nos llevan a estudiar lo ventajoso que pueden ser estos cuerpos.

Definición 2.4 (Primo seudo Mersenne)**.** *Un primo seudo Mersenne es un primo de la forma* $2^n \pm c$ *donde* $\log_2 c \leq \lfloor \frac{1}{2} n \rfloor$.

Definición 2.5 (Cuerpo de Extensión Óptimo)**.** *Un Cuerpo de Extensión Óptimo CEO, está definido como cuerpo finito* $\mathbb{F}_{p^m}$ *tal que* p *es un primo seudo Mersenne y existe un binomio irreducible* $x^m - \omega$ *sobre* $\mathbb{F}_p$.

La aritmética en un CEO es eficiente debido a la eficiencia de la arimética en $\mathbb{F}_p$ y la simplicidad de la reducción módulo el binomio irreducibible $x^m - \omega$ que define $\mathbb{F}_{p^m}$.

Es claro que no todo binomio $x^m - \omega$ será irreducible en $\mathbb{F}_{p^m}$. Si estamos interesados en un CEO especial, es importante analizar si el binomio es irreducible sobre $\mathbb{F}_p$. En [15] se establece el siguiente teorema que proporciona algunas primicias para saber si un binomio de esta forma es irreducible.

Teorema 2.2. *Sea* $m \geq 2$ *un entero y* $\omega \in \mathbb{F}_p$ *. Entonces el binomio* $x^m - \omega$ *es irreducible en* $\mathbb{F}_p$ *si y solo si las siguientes dos condiciones se satisfacen:*

(i). Cada factor primo de m *divide el orden* e *de* ω *en* $\mathbb{F}_p$*, pero no* $\frac{(p-1)}{e}$*;*

(ii). $p \equiv 1 \ mod\ 4$ *si* $m \equiv 0 \ mod\ 4$.

En [4] también se proponen dos casos especiales de CEO los cuales dan ventajas ariméticas adicionales, las cuales son llamadas del Tipo I y Tipo II.

Definición 2.6. *Un CEO de Tipo I tiene como primo* $p = 2^n \pm 1$*, el cual permite una reducción modular eficiente a una baja complejidad.*

Definición 2.7. *Un CEO de Tipo II tiene un binomio irreducible* $x^m - 2$.

Los siguientes son algunos ejemplos de cuerpos de extensión óptimo tipo I y tipo II:

p	f	Parámetros	Tipo
$2^7 + 3$	$x^{13} - 5$	$n = 7,\ c = -3,\ m = 13,\ \omega = 5$	-
$2^{13} - 1$	$x^{13} - 2$	$n = 13,\ c = 1,\ m = 13,\ \omega = 2$	I, II
$2^{31} - 19$	$x^6 - 2$	$n = 31,\ c = 19,\ m = 6,\ \omega = 2$	II
$2^{31} - 1$	$x^6 - 7$	$n = 31,\ c = 1,\ m = 6,\ \omega = 7$	I
$2^{32} - 5$	$x^5 - 2$	$n = 32,\ c = 5,\ m = 5,\ \omega = 2$	II
$2^{57} - 13$	$x^3 - 2$	$n = 57,\ c = 13,\ m = 3,\ \omega = 2$	II
$2^{61} - 1$	$x^3 - 37$	$n = 61,\ c = 1,\ m = 3,\ \omega = 37$	I
$2^{89} - 1$	$x^2 - 3$	$n = 89,\ c = 1,\ m = 2,\ \omega = 3$	I

Tabla 2–8: Cuerpos de Extensión Óptimo tipo I y II

Ejemplo 2.4.1.

Sea $\mathbb{F}_{7^2}$ un cuerpo definido por el binomio irreducible $f(x) = x^2 - 3$. El cuerpo $\mathbb{F}_{7^2}$ está definido como

$$\mathbb{F}_{7^2} = \{a_1\alpha + a_0 \mid a_1,\ a_0 \in \mathbb{F}_7 \text{ y } f(\alpha) = 0\}.$$

Ahora, dado que $f(x) = x^2 - 3$ y $\alpha^2 - 3 = 0$ es decir $\alpha^2 = 3$, entonces los elementos de $\mathbb{F}_{7^2}$ son de la forma

0	$6\alpha+6$	$2\alpha+4$	$\alpha+4$	2α	$5\alpha+1$	$\alpha+5$
$\alpha+6$	5	$2\alpha+2$	$3\alpha+6$	$5\alpha+6$	3α	$4\alpha+5$
$5\alpha+4$	$5\alpha+2$	4	$3\alpha+3$	$\alpha+2$	$4\alpha+2$	α
$6\alpha+4$	$4\alpha+6$	$4\alpha+3$	6	$\alpha+1$	$5\alpha+3$	$6\alpha+3$
5α	$2\alpha+6$	$6\alpha+2$	$6\alpha+1$	2	$5\alpha+5$	$4\alpha+1$
$2\alpha+1$	4α	$3\alpha+2$	$2\alpha+3$	$2\alpha+5$	3	$4\alpha+4$
$6\alpha+5$	$3\alpha+5$	6α	$\alpha+3$	$3\alpha+1$	$3\alpha+4$	1

que son en total 49 elementos.

2.4.1. Adición y Sustracción

Como se indicó al inicio de este capítulo, si $a(x) = \sum_{i=0}^{m-1} a_i x^i$ y $b(x) = \sum_{i=0}^{m-1} b_i x^i$ son elementos de $\mathbb{F}_{p^m}$ entonces

$$a(x) + b(x) = \sum_{i=0}^{m-1} c_i x^i$$

donde $c_i = a_i + b_i \bmod p$. Esto es, se resta p si $a_i + b_i \geq p$. La resta de elementos en $\mathbb{F}_{p^m}$ es análogo, solo que se suma p si $a_i - b_i$ es negativo.

Algoritmo 15 Adición de cuerpo finito

Entrada: $a = a_{m-1}\alpha^{m-1} + \cdots + a_1\alpha + a_0$, $b = b_{m-1}\alpha^{m-1} + \cdots + b_1\alpha + b_0$.
Salida: $a + b \equiv c$

1: **para** $i \leftarrow 0$ to $m-1$ **hacer**
2: $\quad c_i = a_i + b_i$
3: $\quad$ **si** $c_i \geq p$ **entonces**
4: $\quad\quad c_i \leftarrow c_i - p$
5: $\quad$ **fin si**
6: **fin para**

2.4.2. Multiplicación y reducción módulo $x^m - \omega$

La multiplicación en un CEO puede ser expresada vía multiplicación de una matriz muy especial que llamamos matriz modificada de Mastrovito. Esta matriz,

es llamada una matriz de Hankel que es una versión modificada de la matriz de Toeplitz. Estas matrices son interesantes a nivel de hardware porque su estructura permite ser utilizada y aprovechada por métodos como "pipelining", que aumentan el rendimiento de algunos sistemas electrónicos digitales, como microprocesadores [6].

Por otro lado, si $a = a_0 + a_1\alpha + \cdots + a_{m-1}\alpha^{m-1}$ y $b = b_0 + b_1\alpha + \cdots + b_{m-1}\alpha^{m-1}$ son elementos del CEO $\mathbb{F}_{p^m}$, se puede demostrar que el producto $a \cdot b = c = c_0 + c_1\alpha + \cdots + c_{m-1}\alpha^{m-1}$ está dado por

$$\begin{bmatrix} c_{m-1} \\ c_{m-2} \\ \vdots \\ c_1 \\ c_0 \end{bmatrix} = \begin{bmatrix} a_{m-1} & a_{m-2} & a_{m-3} & \cdots & a_1 & a_0 \\ a_{m-2} & a_{m-3} & a_{m-4} & \cdots & a_0 & \omega a_{m-1} \\ & & & \vdots & & \\ a_1 & a_0 & \omega a_{m-1} & \cdots & \omega a_3 & \omega a_2 \\ a_0 & \omega a_{m-1} & \omega a_{m-2} & \cdots & \omega a_2 & \omega a_1 \end{bmatrix} \cdot \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_{m-2} \\ b_{m-1} \end{bmatrix}$$

Ejemplo 2.4.2. *Considere el CEO* $\mathbb{F}_{11^5}$ *definido por el binomio irreducible* $x^5 - 2$ *y sean* $a = 2\alpha^4 + 3\alpha^3 + \alpha^2 + \alpha + 5$ *y* $b = \alpha^3 + 2\alpha + 6$. *Queremos determinar* $a \cdot b$.

La forma usual para calcular el producto de a y b es, primero calcular el producto ordinario de polinomios y luego determinar el residuo módulo el irreducible. En este caso, el producto de polinomios es $c = a \cdot b = 2\alpha^7 + 3\alpha^6 + 5\alpha^5 + 8\alpha^4 + 3\alpha^3 + 8\alpha^2 + 5\alpha + 8$. Entonces, dividiendo este resultado por $\alpha^5 - 2$ obtenemos el residuo $8\alpha^4 + 3\alpha^3 + \alpha^2 + 7$. Ahora usando la definión anterior podemos calcular el producto $a \cdot b$ de la siguiente forma:

$$\begin{bmatrix} c_4 \\ c_3 \\ c_2 \\ c_1 \\ c_0 \end{bmatrix} = \begin{bmatrix} a_4 & a_3 & a_2 & a_1 & a_0 \\ a_3 & a_2 & a_1 & a_0 & 2a_4 \\ a_2 & a_1 & a_0 & 2a_4 & 2a_3 \\ a_1 & a_0 & 2a_4 & 2a_3 & 2a_2 \\ a_0 & 2a_4 & 2a_3 & 2a_2 & 2a_1 \end{bmatrix} \cdot \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix} = \begin{bmatrix} 2 & 3 & 1 & 1 & 5 \\ 3 & 1 & 1 & 5 & 4 \\ 1 & 1 & 5 & 4 & 6 \\ 1 & 5 & 4 & 6 & 2 \\ 5 & 4 & 6 & 2 & 2 \end{bmatrix} \begin{bmatrix} 6 \\ 2 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 8 \\ 3 \\ 1 \\ 0 \\ 7 \end{bmatrix}$$

Por tanto, el resultado final es $a \cdot b = 8\alpha^4 + 3\alpha^3 + \alpha^2 + 7$.

2.4.2.1. Un algoritmo para la multiplicación en un CEO

En la matriz de la definición de la multiplicación, se observa que tiene diagonales secundarias constantes. Así que es necesario generar solo la primera fila dado que cada fila despúes de la primera puede ser generada al multiplicar el primer elemento de la fila actual por ω y luego hacer un desplazamiento circular por izquierda. Esto nos lleva a un algoritmo similar al algoritmo de la multiplicación en $\mathbb{F}_{2^m}$ desarrollado en [10]. Una diferencia es que ahora cada segmento de a, b, c y la fila S de la matriz, consiste de un elemento de $\mathbb{F}_p$,i.e.,$\lceil \log_2 p \rceil$ en vez de un solo bit [17].

Algoritmo 16 Multiplicación en CEO

Entrada: $A(\alpha), B(\alpha), p$ los coeficientes de A son dados en el orden $a_{m-1}, a_{m-2}, \cdots, a_0$ y similarmente para B.
Salida: $C(\alpha)$ inicialización $S \leftarrow A$;

1: **para** $i \leftarrow 0$ to $m-1$ **hacer**
2: $\quad c_{m-i-1} \leftarrow S \cdot B \, mod \, p$
3: $\quad S_{m-1} \leftarrow \omega * S_{m-1}$
4: $\quad S \leftarrow S << \lceil \log_2 p \rceil$
5: **fin para**

2.4.3. Inversión

El siguiente es un método para el cálculo directo de un inverso $b(\alpha) = a^{-1}(\alpha)$ en $\mathbb{F}_{p^m}$. Se considera el producto $c(\alpha) = a(\alpha)b(\alpha) = 1 \bmod f(\alpha)$, donde $f(\alpha)$ denota el polinomio irreducible que define cuerpo. Esto significa que el primer coeficiente de el producto $a(\alpha)\, b(\alpha)$ es uno y todos los demas coeficientes son ceros. Expresando los coeficientes de el producto en términos de los coeficientes de a y b, se forma un sistema de m ecuaciones lineales y resolviendo estas ecuaciones para los coeficientes de b queda el inverso expresado en términos de los coeficientes de a [5] .

La ventaja principal de usar la técnica de inversión directa es que se puede calcular el inverso de un elemento en una extensión de cuerpo, haciendo operaciones solo en el cuerpo $\mathbb{F}_p$ y cuando $m = 2$ o $m = 3$ podemos expresar los inversos en términos de fórmulas como sigue a continuación.

Los siguientes dos ejemplos ilustran los casos mencionados anteriormente para m.

Ejemplo 2.4.3. *Inversión directa en* $\mathbb{F}_{p^2}$

Sea $a(\alpha) \in \mathbb{F}_{p^2}$ y $a(\alpha) = a_1\alpha + a_0$, donde $a_0, a_1 \in \mathbb{F}_p$, con el siguiente polinomio irreducible $f(x) = x^2 - \omega$, donde $\omega \in \mathbb{F}_p$. Entonces, el inverso $b(\alpha) = b_1\alpha + b_0$ tiene que satisfacer

$$\begin{aligned} a(\alpha)b(\alpha) \bmod f(x) &= (a_1\alpha + a_0) \cdot (b_1\alpha + b_0) \bmod f(x) \\ &= (\alpha a_1 b_1 + a_0 b_0) + (a_1 b_0 + a_0 b_1)\alpha \\ &= 1 \end{aligned}$$

Los coeficientes producen el siguiente sitema de ecuaciones

$$\begin{pmatrix} a_1 & a_0 \\ a_0 & \omega a_1 \end{pmatrix} \begin{pmatrix} b_0 \\ b_1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

Resolviendo el sistema de ecuaciones,

$$b_0 \;=\; a_0\Delta^{-1} \quad \text{y } b_1 = a^{-1}\Delta^{-1} \text{ donde } \Delta = a_0^2 - \omega\, a_1^2.$$

Ejemplo 2.4.4. *Inversión directa en* $\mathbb{F}_{p^3}$

Sea $a(\alpha) = a_2\alpha^2 + a_1\alpha + a_0 \in \mathbb{F}_{p^3}$ definido por el binomio irreducible $f(x) = x^3 - \omega$. Entonces, los coeficientes del inverso $b(\alpha) = b_2\alpha^2 + b_1\alpha + b_0$ satisfacen el siguiente sistema de ecuaciones:

$$\begin{pmatrix} a_2 & a_1 & a_0 \\ a_1 & a_0 & \omega a_2 \\ a_0 & \omega\, a_2 & \omega a_1 \end{pmatrix} \begin{pmatrix} b_0 \\ b_1 \\ b_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

Resolviendo el sistema de ecuaciones se obtiene,

$$\begin{aligned} b_0 &= (a_0^2 - a_1\, a_2\, \omega)\Delta^{-1} \\ b_1 &= (a_2^2\omega - a_0\, a_1)\Delta^{-1} \\ b_2 &= (a_1^2 - a_0\, a_2)\Delta^{-1} \end{aligned}$$

donde $\Delta = a_0^3 - 3a_0a_1a_2\omega + a_1^3\omega + a_2^3\omega^2$.

2.4.4. Exponenciación cuadrática

La exponenciación cuadrática puede ser implementada usando el método general para computar una multiplicación. Sin embargo, es posible que algunos cuerpos posean características especiales que permiten que esta operación se pueda resolver de manera más eficiente a la tradicional [4]. Por ejemplo, si consideramos el elemento

$a(\alpha) = a_2\alpha^2 + a_1\alpha + a_0$ en el cuerpo $\mathbb{F}_{p^3}$ definido por un binomio irreducible $x^3 - 2$, nosostros podemos computar el cuadrado de $a(\alpha)$ de la siguiente forma:

$$(a(\alpha))^2 = (a_2a_0 + a_1^2 + a_0a_2)\alpha^2 + (a_1a_0 + a_0a_1 + 2a_2^2)\alpha + (a_0^2 + 2a_2a_1 + 2a_1a_2)$$

que simplificando obtenemos la siguiente expresión

$$(a(\alpha))^2 \quad = \quad (2a_2a_0 + a_1^2)\alpha^2 + (2a_2^2 + 2a_1a_0)\alpha + (4a_2a_1 + a_0^2)$$

Esta nueva expresión nos muestra que solo debemos hacer 6 multiplicaciones y 5 desplazamientos en comparación con el método tradicional en el que hariamos 9 multiplicaciones.

2.4.5. Cuerpos de Extensión Óptimo para FPGA

Anteriormente observamos que los primos seudo Mersenne $p = 2^n - c$ son frecuentemente usados en criptografía de clave pública porque admiten una rápida reducción modular.

Si x es un entero positivo menor que p^2, entonces x puede ser escrito como

$$x = x_2 \cdot 2^{2n} + x_1 \cdot 2^n + x_0$$

donde x_2 es 0 o 1 y $x_1,\ x_0$ representan bloques de 2^n, entonces

$$x \equiv x_2 \cdot c^2 + x_1 \cdot c + x_0 \bmod p$$

y luego después de a lo sumo 2 sustracciones de p, el valor de x será reducido módulo p [27]. Esta reducción funciona bastante bien en software, especialmente en computadoras con un largo de palabra fijo. Pero la multiplicación estándar en FPGAs puede ser costosa. Por esto introducimos una variante de CEOs que es más apropiado para hardware.

Definición 2.8. *Un Cuerpo de Extensión Óptimo para FPGAs (CEOFPGA), es un cuerpo* $\mathbb{F}_{p^m}$ *con las siguientes propiedades:*

1. p *es de la forma* $2^n - (2^i + 1)$ *con* $i < n$
2. $\mathbb{F}_{p^m}$ *se define por un binomio* $x^m - \omega$ *irreducible modulo* p

Se definen dos tipos de CEOFPGAs:

- Tipo I: ω es una potencia de 2
- Tipo II: ω no es una potencia de 2

Con tal tipo de primo la reducción módulo p puede ser llevada a cabo con un mínimo número de desplazamientos y sumas. Si el CEOFPGA es de tipo I, la reducción módulo el binomio irreducible también puede ser llevada a cabo mediante desplazamientos y sumas. Entonces la pregunta es ¿Dado un entero positivo q, existe un CEOFPGA $\mathbb{F}_{p^3}$ tal que p^3 es igual a (o aproximadamente igual a) q?

n	i	ω	Caso NIST [21]
54	5	2	$\mathbb{F}_{2^{163}}$
77	–	–	$\mathbb{F}_{2^{233}}$
94	13	2	$\mathbb{F}_{2^{283}}$
136	47	2	$\mathbb{F}_{2^{409}}$
64	29	2	$\mathbb{F}_{\bar{p}}$ $\|\bar{p}\|$=192
74	25	2	$\mathbb{F}_{\bar{p}}$ $\|\bar{p}\|$=224
85	–	–	$\mathbb{F}_{\bar{p}}$ $\|\bar{p}\|$=256
128	–	–	$\mathbb{F}_{\bar{p}}$ $\|\bar{p}\|$=384
173	–	–	$\mathbb{F}_{\bar{p}}$ $\|\bar{p}\|$=521

Tabla 2–9: CEOFPGAs comparados con cuerpos Binarios y Primos

El símbolo $\|\bar{p}\|$ indica la cantidad de "bits" de $\bar{p}$ y los espacios que aparecen en blanco indican que hasta el momento no se ha podido encontrar CEOFPGAs de tipo I que sean comparables a los casos NIST, por esa razón resultaría interesante establecer condiciones necesarias y suficientes, para que dado un número positivo q exista un CEOFPGA.

2.5. Multiplicación y reducción sobre $\mathbb{F}_{239^{17}}$

Nosotros implementamos la multiplicación en el cuerpo $\mathbb{F}_{239^{17}}$ definido por el binomio irreducible $x^{17} - 2$. Representamos cada elemento $a(\alpha) = a_{16}\alpha^{16} + \cdots + a_0$ por un vector de los coeficientes $(a_{16}, \ldots, a_0)$ donde cada a_i es un elemento de 8 "bits" en $\mathbb{F}_{239}$.

El circuito consiste de 3 componentes: un multiplicador paralelo que computa los productos en cada producto interior, un sumador "pipelined" que suma estos productos y un reductor módulo 239 que reduce estas sumas módulo 239.

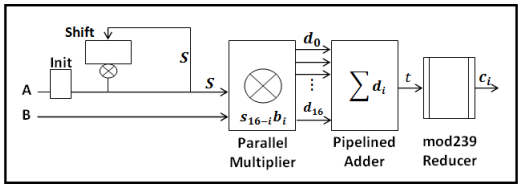


Figura 2–1: Multiplicador en $\mathbb{F}_{239^{17}}$

El orden para calcular el producto $c(\alpha) = a(\alpha) \cdot b(\alpha)$ de dos elementos en $\mathbb{F}_{239^{17}}$ es como sigue:

- Comenzando en un estado inicial, asignamos $(a_{16}, a_{15}, \ldots, a_1, a_0)$ a la señal S, la cual también sirve como una entrada en el multiplicador paralelo y el componente "shift".
- Las señales S y b (i.e., $(b_{16}, b_{15}, \ldots, b_1, b_0)$) en el multiplicador paralelo calculan los productos $d_i = a_{16-i}$ en paralelo.
- Cuando la señal S llega al componente "shift" , el producto mod 239 de $2 \cdot a_{16}$ es computado y S es cíclicamente desplazado en 8 "bits" a la izquierda y se repiten los pasos de arriba con un nuevo valor de S.

2.5.1. Componente mod 239

Nosotros también usamos una idea presentada en [4] para simplificar la operación redución mod 239, una operación muy costosa. Este resultado de computar

$S \cdot B$ es c, entonces $c = c[2] * 2^{16} + c[1] * 2^8 + c[0]$ donde cada $c[i] < 256$. Si usamos el hecho de que $2^{16} = 50 \bmod 239$ y $2^8 = 17 \bmod 239$ para reducir un producto interno como sigue:

$$\begin{aligned} c &\leftarrow c[2] * 50 + c[1] * 17 + c[0] \\ c &\leftarrow c[1] * 17 + c[0] \\ Si \quad c > 239 &\quad \text{entonces } c \leftarrow c - 239 \\ Si \quad c > 239 &\quad \text{entonces } c \leftarrow c - 239 \end{aligned}$$

2.5.2. Resultados experimentales

Nosotros programamos el algoritmo 16 en VHDL. Nuestra tarjeta es Xilinx Spartan 3-speed(-5). Usamos la herramienta Xilinx XST (Xilinx Synthesis technology) incluido el paquete ISE Project Navigator para simular nuestro programa. La siguiente tabla compara el rendimiento de nuestro algoritmo para las implementaciones de Deschamps et al [7], en el mismo hardware de dos algoritmos bien conocidos como los algoritmos MSE ("Most Significant Element") y el LSE ("Least Significant Element"). Después cada programa usa 3 paquetes generales. La generalidad de estas tres operaciones cuenta para la diferencia en el rendimiento.

	FFs	**LUTs**	**Slices**	**Tiempo total (ns)**
MSE- first	303	1,690	937	888
LSE-first	415	1,779	1,061	330
Nuestro Algoritmo	304	1,306	851	216

Tabla 2–10: Multiplicador en $\mathbb{F}_{239^{17}}$

La tabla anterior nos indica que nuestro algoritmo gana tanto en tiempo como en área con respecto a las implementaciones presentadas en [7]. Sin embargo, podríamos

ganar aún más tiempo, si utilizaramos el algoritmo 9, en vez de la multiplicación proporcionada por VHDL.

Capítulo 3
CURVAS ELÍPTICAS SOBRE CUERPOS FINITOS

En este capítulo veremos algunos conceptos generales acerca de curvas elípticas definidas sobre cuerpos finitos. Detallaremos algunas nociones matemáticas sobre las operaciones del grupo que define una curva elíptica como lo son la ley de grupo y la multiplicación escalar, viendo esta última como una operación esencial para los criptosistemas de curva elíptica.

3.1. Conceptos generales

Las curvas elípticas son estructuras muy interesantes, la fascinación por ellas radica en las propiedades que estas presentan y lo útiles que pueden ser en la seguridad informática.

A continuación, definamos formalmente y detallemos algunos aspectos de lo que denotan estos objetos.

Definición 3.1 (**Curva elíptica**). *Una curva elíptica E sobre un cuerpo $\mathbb{K}$ está definida por una ecuación de Weierstrass de la forma:*

$$E : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \tag{3.1}$$

donde $a_1, a_2, a_3, a_4, a_6 \in \mathbb{K}$ *y* $\Delta \neq 0$, *donde* Δ *es el discriminante de* E *y es de la siguiente forma:*

$$\begin{aligned} \Delta &= -d_2^2 d_8 - 8d_4^3 - 27d_6^2 + 9d_2 d_4 d_6 \\ d_2 &= a_1^2 + 4a_2 \\ d_4 &= 2a_4 + a_1 a_3 \\ d_6 &= a_3^2 + 4a_6 \\ d_8 &= a_1^2 a_6 + 4a_2 a_6 - a_1 a_3 a_4 + a_2 a_3^2 - a_4^2 \end{aligned}$$

Entonces, el conjunto de puntos sobre E es :

$$E(\mathbb{K}) = \{(x, y) \in \mathbb{K} \times \mathbb{K} : y^2 + a_1 xy + a_3 y - x^3 - a_2 x^2 - a_4 x - a_6 = 0\} \cup \{\infty\}$$

Donde ∞ es el punto infinito, del que daremos su significado más adelante. [26]

Los grupos de curvas elípticas son grupos aditivos [26], es decir la operación básica es la suma. Empecemos ilustrando la idea para curvas sobre los reales, las operaciones entre puntos sobre una curva elíptica definida sobre los reales pueden ser representadas de manera gráfica.

Si P y Q son dos puntos distintos en una curva elíptica podemos calcular las suma de estos puntos obteniendo como resultado un tercer punto R. Esta operación es denotada como $R = P + Q$.

El inverso de un punto P se obtiene haciendo una reflexión alrededor de el eje x. Es decir, si $P = (x, y)$ entonces $-P = (x, -y)$.

Para sumar los puntos P y Q se traza una línea a través de los puntos como se muestra en la siguiente figura, esta línea corta a la curva en un tercer punto llamado

$-R$, y luego este es reflejado con respecto al eje x para obtener el punto R que es el punto resultante de la suma entre P y Q. En la figura 3.1 podemos observar una representación geométrica de la suma de dos puntos diferentes en una curva elíptica.

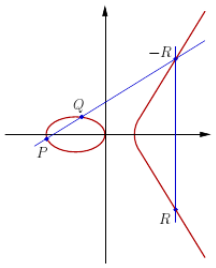


Figura 3–1: Suma de puntos $R = P + Q$

Existe un caso especial en la suma cuando queremos sumar P con $-P$, debido a que la línea que forman estos puntos nunca corta a la curva elíptica en un tercer punto. Es por esta razón que el grupo de la curva incluye el elemento infinito ∞. Por definición $P+(-P) = \infty$ y como resultado de esta ecuación tenemos que $P+\infty = P$ en el grupo, este caso es mostrado en la figura 3-2.

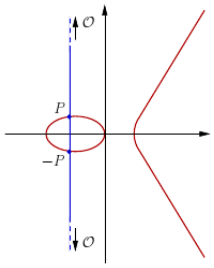


Figura 3–2: Suma de puntos $P + (-P) = \infty$

La operación de doblado de un punto es denotado por $2P = P + P = R$ y es mostrada en la figura 3-3, esta operación consiste en sumar el punto P consigo mismo. De manera gráfica se traza una línea tangente a la curva que toca el punto P y si se cumple que la coordenada del punto es diferente de ∞, entonces la tangente corta a la curva elíptica en otro punto $-R$, luego este es reflejado con respecto al eje x y se obtiene así el punto R. Sin embargo, cuando doblamos un punto P y la coordenada y es cero entonces la tangente a la curva que toca este punto nunca corta

a la curva en algún otro punto, por esta razón el resultado es ∞. En la figura 3-4 se puede observar este caso.

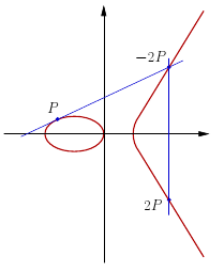


Figura 3–3: Suma de puntos $2P = P + P = R$

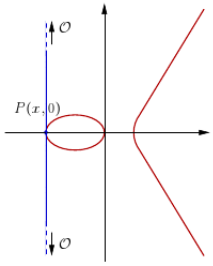
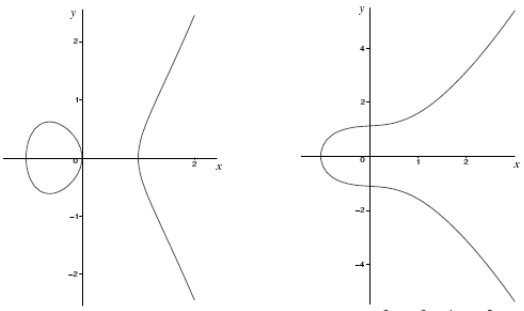


Figura 3–4: Suma de puntos $2P = \infty$ *si* $P = (x, 0)$

Ejemplo 3.1.1 (Curvas elípticas sobre $\mathbb{R}$)**.** *Consideremos las curvas elípticas*

$$E_1 : y^2 = x^3 - x \qquad E_2 : y^2 = x^3 + \frac{1}{4}x + \frac{5}{4}$$

definidas sobre el cuerpo $\mathbb{R}$ *de los números reales.*



(a) $E_1 : y^2 = x^3 - x$ (b) $E_2 : y^2 = x^3 + \frac{1}{4}x + \frac{5}{4}$

Figura 3–5: $E_1 : y^2 = x^3 - x$ $E_2 : y^2 = x^3 + \frac{1}{4}x + \frac{5}{4}$ sobre $\mathbb{R}$

En el ejemplo anterior, las gráficas de E_1 y E_2 muestran cierta suavidad al ser expresadas en el plano, esto se debe al hecho de que las curvas están definidas sobre el cuerpo de los reales. Si una curva está definida sobre un cuerpo finito, entonces el conjunto solución para la curva también será finito y por tanto esto nos lleva a pensar que la gráfica que obtendremos será un conjunto de puntos dispersos en el plano.

Por ejemplo, la siguiente gráfica muestra una curva elíptica E definida sobre el

cuerpo finito $\mathbb{F}_{29}$

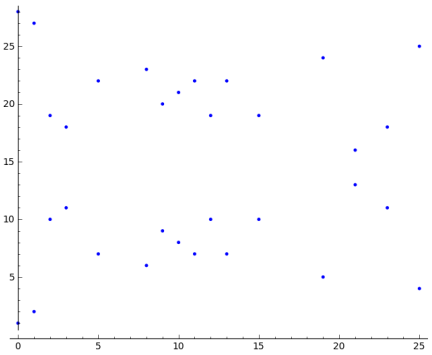


Figura 3–6: $E : y^2 = x^2 + 2x + 1$ sobre $\mathbb{F}_{29}$

También se observa la simetría que presenta la gráfica de E, dado que la ecuación que define la curva tiene al menos dos soluciones para cada $x \in \mathbb{F}_{29}$.

Veamos ahora, como se definen curvas elípticas sobre cuerpos infinitos o finitos y de las variaciones que están pueden tener dependiendo de la característica del cuerpo.

3.1.1. Representación de una curva elíptica

Una ecuación de Weierstrass

$$E : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$$

definida sobre un cuerpo finito $\mathbb{K}$, puede ser simplificada considerablemente si aplicamos algunos cambios admisibles a las variables. Consideremos por separado los casos donde el cuerpo $\mathbb{K}$ tiene característica diferente de 2 o 3. [11]

1. Si la característica de $\mathbb{K}$ no es igual a 2 o 3, entonces el cambio admisible de variables

$$(x, y) \to \left(\frac{x - 3a_1^2 - 12a_2}{36}, \frac{y - 3a_1x}{216} - \frac{a_1^3 + 4a_1a_2 - 12a_3}{24} \right)$$

transforma E a la curva

$$y^2 = x^3 + ax + b$$

donde $a, b \in \mathbb{K}$. El discriminante de esta curva es $\Delta = -16(4^3 + 27b^2)$.

2. Si la característica de $\mathbb{K}$ es 2, entonces hay dos casos a considerar. Si $a_1 \neq 0$, entonces el cambio admisible de variables

$$(x, y) \to \left(a_1^2 x + \frac{a_3}{a_1}, a_1^3 y + \frac{a_1^2 a_4 + a_3^2}{a_1^3} \right)$$

transforma E a la curva

$$y^2 + xy = x^3 + ax^2 + b$$

donde $a, b \in \mathbb{K}$. Tal curva se le llama no-supersingular y tiene discriminante $\Delta = b$.

Si $a_1 = 0$, entonces el cambio admisible de variables

$$(x, y) \to (x + a_2, y)$$

transforma E a la curva

$$y^2 + cy = x^3 + ax + b$$

donde $a, b, c \in \mathbb{K}$. Tal curva se le llama supersingular y tiene discriminante $\Delta = c^4$.

3. Si la característica de $\mathbb{K}$ es 3, entonces hay dos casos a considerar.

Si $a_1^2 \neq -a_2$, entonces el cambio admisible de variables

$$(x, y) \to \left(x + \frac{d_4}{d_2}, y + a_1 x + a_1 \frac{d_4}{d_2} + a_3 \right),$$

donde $d_2 = a_1^2 + a_2$ y $d_4 = a_4 - a_1 a_3$ transforma E a la curva

$$y^2 = x^3 + ax^2 + b$$

donde $a, b \in \mathbb{K}$. Tal curva se dice no-supersingular y su discriminante es $\Delta = -a^3 b$.

Si $a_1^2 = -a_2$, entonces el cambio admisible de variables $(x, y) \to (x, y + a_1 x + a_3)$ transforma E a la curva

$$y^2 = x^3 + ax + b$$

donde $a, b \in \mathbb{K}$. Tal curva es llamada supersingular y tiene discriminante $\Delta = -a^3$.

Como habíamos mencionado anteriormente, en particular se puede hacer del conjunto $E(\mathbb{K})$ un *grupo*, es decir proveer al conjunto de una operación binaria y de un elemento identidad.

Veamos entonces a continuación las operaciones que se pueden establecer en el grupo que define una curva elíptica.

3.2. Operaciones de curva elíptica

Los grupos de curvas elípticas son grupos aditivos, es decir, la operación básica es la suma. Esta operación tiene una definición peculiar que estaremos definiendo en detalle en la sección de ley de grupo y es la que permite que el conjunto $E(\mathbb{K})$ sea un grupo, ya que cumple con las propiedades conmutativa, asociativa, de identidad y existencia de un inverso [26].

Por otro lado, la operación de multiplicación de dos puntos en un grupo aditivo de curvas elípticas no existe, sin embargo, la multiplicación de un escalar por un punto en la curva sí y esta puede ser obtenida mediante la suma repetitiva del mismo punto.

Definamos inicialmente mediante la ley de grupo como sumar dos puntos de una curva elíptica E.

3.2.1. Ley de grupo

Sea E una curva elíptica definida por $y^2 = x^3 + ax + b$ sobre un cuerpo $\mathbb{K}$ con característica distinta de 2 y 3. Sea $P = (x_1, y_1)$ y $Q = (x_2, y_2)$ puntos sobre E con $P, Q \neq \infty$. Definimos $P + Q = (x_3, y_3)$ como sigue:

1. Si $x_1 \neq x_2$ entonces

$$x_3 = m^2 - x_1 - x_2, \qquad y_3 = m(x_1 - x_3) - y_1 \quad \text{donde} \quad m = \frac{y_2 - y_1}{x_2 - x_1}$$

2. $x_1 = x_2$ pero $y_1 \neq y_2$ entonces $P + Q = \infty$
3. Si $P = Q$ y $y_1 \neq 0$ entonces

$$x_3 = m^2 - 2x_1, \qquad y_3 = m(x_1 - x_3) - y_1 \quad \text{donde} \quad m = \frac{3x_1^2 + a}{2y_1}$$

4. Si $P = Q$ y $y_1 = 0$, entonces $P + Q = \infty$ más aún $P + \infty = P$, para todos los puntos $P \in E$.

Note que cuando P y Q tienen coordenadas en un cuerpo $\mathbb{K}$, entonces $P + Q$ también tiene coordenadas en $\mathbb{K}$. Por consiguiente, $E(K)$ es cerrado bajo la adición. [26]

La definición anterior muestra como sumar dos puntos en una curva elíptica, sin embargo ahora estamos interesados en determinar el producto de un escalar por un punto elíptico, el interés de esta operación reside en el hecho de que es una de las operaciones principales en los criptosistemas de curva elíptica y una de las operaciones que más se desea optimizar en esta área. En la siguiente sección, se muestra como determinar esta operación mediante uno de los métodos más conocidos el método binario, que recibe su nombre debido a que hace uso de la representación binaria de el escalar que multiplica al punto elíptico.

3.2.2. Multiplicación escalar

Como mencionamos anteriormente, la multiplicación de dos puntos en un grupo definido por una curva elíptica no existe, sin embargo la multiplicación escalar kP donde $k \in \mathbb{Z}^+$ puede ser obtenida mediante la suma de k veces el mismo punto P , esto es

$$kP = \underbrace{P + P + ... + P}_{k-veces}$$

Uno de los métodos para realizar esta operación consiste en utilizar la suma y doblado de puntos. Dado que el conjunto $E(\mathbb{K})$ puede ser un grupo, con la definición anterior de multiplicación escalar podemos entonces introducir el término de orden de un punto en una curva elíptica.

Definición 3.2 (Orden de un punto)**.** *El orden de un punto P sobre una curva E es el entero positivo n más pequeño tal que $nP = \infty$.*

Veamos ahora un ejemplo del producto escalar sobre una curva E definida en el cuerpo $\mathbb{F}_{7^2}$

Ejemplo: Producto escalar sobre $E(\mathbb{F}_{7^2})$. Note que los elementos de este cuerpo fueron definidos anteriormente en el ejemplo 2.4.1.

Sea E una curva elíptica definida por $y^2 = x^3 + (\alpha+1)x + 2\alpha$ sobre el cuerpo $\mathbb{F}_{7^2}$. Si tomamos $x = 0$ entonces $y^2 = 2\alpha$, por tanto debemos buscar si es posible dos soluciones y_1 y y_2 tales que $y_i^2 = 2\alpha$. En efecto si tomamos $y_1 = (5\alpha + 3)$ entonces $y_1^2 = 25\alpha^2 + 30\alpha + 9 \equiv 4(3) + 2\alpha + 2 \equiv 2\alpha$ por tanto y_1 es una solución, así mismo, podemos verificar para $y_2 = 2\alpha + 4$. Los siguientes elementos fueron generados usando el paquete matemático SAGE [24]. El grupo generado por esta curva es:

∞	$(0, 5\alpha+3)$	$(4, \alpha+3)$	$(4, 6\alpha+4)$	$(5, 2)$
$(5, 5)$	$(6, \alpha+4)$	$(6, 6\alpha+3)$	$(\alpha+1, \alpha+5)$	$(\alpha+1, 6\alpha+2)$
$(\alpha+5, 3\alpha+5)$	$(\alpha+5, 4\alpha+2)$	$(2\alpha, 3\alpha)$	$(2\alpha, 4\alpha)$	$(2\alpha+1, 3)$
$(2\alpha+1, 4)$	$(2\alpha+2, \alpha+6)$	$(2\alpha+2, 6\alpha+1)$	$(2\alpha+6, 3\alpha+2)$	$(2\alpha+6, 4\alpha+5)$
$(3\alpha+5, \alpha+4)$	$(3\alpha+5, 6\alpha+3)$	$(4\alpha+3, \alpha+4)$	$(4\alpha+3, 6\alpha+3)$	$(5\alpha, 3\alpha+3)$
$(5\alpha, 4\alpha+4)$	$(5\alpha+2, 2\alpha+6)$	$(5\alpha+2, 5\alpha+1)$	$(6\alpha, \alpha+6)$	$(6\alpha, 6\alpha+1)$
$(6\alpha+1, 3\alpha+4)$	$(6\alpha+1, 4\alpha+3)$	$(6\alpha+2, \alpha+1)$	$(6\alpha+2, 6\alpha+6)$	$(6\alpha+5, \alpha+6)$
$(6\alpha+5, 6\alpha+1)$	$(6\alpha+6, 3\alpha+6)$	$(6\alpha+6, 4\alpha+1)$	$(0, 2\alpha+4)$	

con un total de 39 elementos.

Ahora, nos interesa calcular el punto $4P$ donde $P = (2\alpha, 3\alpha)$. Por el método de doblado de puntos, podemos expresar $4P$ como $4P = 2(2P)$ y según la ley de grupo tenemos que $2P = P + P$ luego

$$\begin{aligned} m &= \tfrac{3(2\alpha)^2+\alpha+1}{2(3\alpha)} = (16+\alpha)(2(3\alpha))^{-1} \\ &= (\alpha+2)(2\alpha) = 2\alpha^2 + 4\alpha \\ &= 2\cdot 3 + 4\alpha = 4\alpha + 6 \\ x_3 &= m^2 - 2(2\alpha) = 6\alpha - 4\alpha \\ &= 2\alpha \\ y_3 &= m(2\alpha - 2\alpha) - (3\alpha) \\ &= 4\alpha \end{aligned}$$

Si denotemos $Q = (2\alpha, 4\alpha)$ y aplicando nuevamente la ley de grupo para $2Q$ tenemos que $2Q = (2\alpha, 3\alpha)$, osea que $4P = 2(2P) = 2Q = (2\alpha, 3\alpha)$. Note que P y $4P$ son iguales, es decir

$$\begin{aligned} 4P &= P \\ 3P + P &= P \\ 3P &= P + (-P) \\ 3P &= \infty \end{aligned}$$

Esto quiere decir que el orden de P es 3.

El siguiente método es similar al anterior, este también hace uso de la suma y doblado de puntos, solo que depende de la representación binaria de del valor de k. Veamos el siguiente algoritmo que representa otra forma de hacer multiplicación escalar en $E(\mathbb{K})$ Observemos un ejemplo para ilustrar esta idea.

Ejemplo 3.2.1. *Sea E una curva y P un punto sobre ella, determinemos $21P$ usando el algoritmo anterior.*

Observe que en $i = 4$ se obtiene el valor de $21P$.

Algoritmo 17 Multiplicación escalar en $E(\mathbb{K})$

1: **Entrada:** $P \in E(\mathbb{K})$ y $k = (k_{t-1}, ..., k_1, k_0)_2$ (Representación binaria)
2: **Salida:** kP
3: $Q \leftarrow \infty$
4: Para i hasta $t-1$ haga
   Si $k_i = 1$ entonces $Q \leftarrow Q + P$
5: $P \leftarrow 2P$
6: Retorne Q

i	k_i	Q	P
		∞	P
0	1	P	$2P$
1	0	P	$4P$
2	1	$5P$	$8P$
3	0	$5P$	$16P$
4	1	$21P$	$32P$

Tabla 3–1: Multiplicación escalar para $21P$

Por otro lado, dado que el conjunto de puntos racionales que define una curva E sobre un cuerpo finito es finito, es posible hablar del cardinal del grupo que define una curva. En ocasiones, también se dice orden de una curva definida sobre un cuerpo. [26]

Definición 3.3 (Orden del grupo)**.** *El orden de una curva elíptica E definida sobre el cuerpo $\mathbb{K}$ es el número de puntos sobre la curva elíptica E, incluyendo el punto ∞ y se denota por $\#E(\mathbb{K})$.*

El teorema de Hasse [26], proporciona límites más estrictos para $\#E(\mathbb{K})$

Teorema 3.1 (Hasse). *Sea E una curva elíptica definida sobre $\mathbb{K}$ con característica q, entonces*

$$q + 1 - 2\sqrt{q} \leq \#E(\mathbb{K}) \leq q + 1 + 2\sqrt{q}.$$

El intervalo $[q + 1 - 2\sqrt{q}, q + 1 + 2\sqrt{q}]$ es llamado el intervalo de Hasse.

Como se ha indicado anteriormente, los cuerpos finitos son estructuras algebraicas que por su naturaleza, nos habilitan para hacer criptografía a diferencia de cuerpos como los reales, los cuales no se recomiendan por motivos de truncamiento y redondeo.

Los cuerpos binarios y primos son populares en la criptografía de curva elíptica. En [21] se discuten algunos de los parámetros de dominio de curvas elípticas, estos parámetros están intimamente relacionados con la escogencia de los cuerpos finitos en los cuales se puede definir la curva, lo que nos lleva a pensar que también han tomado en cuenta la importancia de la aritmética sobre el cuerpo. Los Cuerpos de Extensión Óptimo, son cuerpos que proporcionan facilidades en su aritmética, por lo que sería interesante comparar dichos cuerpos con algunos casos ya conocidos como cuerpos $\mathbb{F}_{2^{163}}$.

En el siguiente capítulo, trataremos de responder a este interrogante. Se presentaran algunos resultados de implementaciones con los que podremos comparar y concluir algunas de las ventajas y desventajas al utilizar estos tipos de cuerpos.

Capítulo 4
IMPLEMENTACIÓN Y RESULTADOS

En el siguiente capítulo se describen implementaciones y resultados que muestran el rendimiento de las operaciones sobre el cuerpo de extensión óptimo $\mathbb{F}_{p^m}$ con $p = 2^{54} - 33$ y $m = 3$, definido por el binomio irreducible $x^3 - 2$, con el proposito de mostrar a los cuerpos de extensión óptimo como otra alternativa viable y eficiente en el rendimiento de operaciones de curva elíptica definidas en estos cuerpos, en comparación a el rendimiento que se obtiene con los cuerpos de mayor uso como los cuerpos primos y binarios. En particular, se describe una implementación acerca de la multiplicación escalar para una curva elíptica definida sobre este mismo cuerpo. Los diseños de estas operaciones fueron implementadas en VHDL[22] y fueron sintetizadas y simuladas para el dispositivo FPGA de Xilinx (Spartan3 Speed-5).

4.1. Cuerpos de extensión óptimo para curvas elípticas

En los capítulos anteriores se ha intentado mostrar la importancia de tener una aritmética eficiente sobre el cuerpo finito en el cual esté definida una curva elíptica.

En general, en criptografía el estudio sobre elegir cuerpos finitos adecuados que permitan a los criptosistemas basados en curvas elípticas tener un desempeño eficiente, en su mayor parte, lo han recibido los cuerpos binarios y cuerpos primos.

No es de extrañarse que estos cuerpos tengan un especial interés, pues se ha mostrado que los cuerpos binarios son eficientes a nivel de hardware por la naturaleza en su aritmética y que los cuerpos primos evitan la reducción de un polinomio

irreducible a diferencia de otros cuerpos finitos. Sin embargo, en los cuerpos binarios existe el inconveniente, que la reducción módulo el polinomio irreducible puede ser costosa si el grado del polinomio es muy grande y por otro lado los cuerpos primos también presentan el inconveniente de que las operaciones como la multiplicación pueden ser altamente costosas a nivel de hardware si el tamaño en "bits" de los elementos es grande. Estos inconvenientes nos motivan a proponer en este documento otro tipo de cuerpos, que podrían solucionar algunos de los problemas antes mencionados: Los cuerpos de extensión óptimo.

Como se definió en el capítulo 2, la aritmética en los cuerpos de extensión óptimo $\mathbb{F}_{p^m}$ es eficiente debido a la simplicidad módulo el binomio irreducible que define el cuerpo y a la eficiente aritmética en el cuerpo $\mathbb{F}_p$, ya que la forma del seudoprimo p permite que la reducción modular sea más simple. Por ejemplo, en [7, 21] estudian estructuras como el cuerpo binario $\mathbb{F}_{2^m}$ con $m = 163$ y el cuerpo primo $\mathbb{F}_p$ donde p es un primo de 160 "bits", nos gustaría construir un cuerpo de extensión óptimo que en cierta forma se aproxime a la cantidad de "bits" basadas en cada cuerpo y así poder hacer algunas comparaciones y conclusiones en rendimiento. Una opción puede ser el cuerpo $\mathbb{F}_{p^m}$ con $p = 2^{54} - 33$ y $m = 3$, definido por el binomio irreducible $x^3 - 2$. El tamaño en número de "bits" de un elemento en este cuerpo visto como un vector, (lo cual resulta adecuado para representación en hardware), sería de 162 "bits", un promedio que nos motiva a comparar y estudiar las propiedades de cada uno de estos cuerpos. Primero, mientras que las multiplicaciones ordinarias en el cuerpo primo se hacen con elementos de 160 "bits" en el cuerpo de extensión óptimo se hacen con elementos de 54 "bits" lo cual resulta bastante ventajoso. Segundo, mientras la reducción módulo un polinomio irreducible en $\mathbb{F}_{2^{163}}$ se hace para un polinomio de grado 163, en el CEO la reducción se hace con un grado muchisimo menor, además de que el polinomio irreducible resulta ser un binomio, lo que hace

que la reducción sea más simple.

La reducción módulo el primo en el CEO y cuerpo primo es otro caso que hay que estudiar, esto depende de la forma que tenga cada primo, ya que el hecho de tener un seudoprimo de la forma $p = 2^n - c$ puede significar una disminución en multiplicaciones y sumas en la reducción modular, a diferencia de otros que no posean una forma peculiar.

Por otro lado, dado que la multiplicación escalar es una de las operaciones esenciales y de frecuente uso en los esquemas criptográficos de curva elíptica, siempre se ha buscado optimizar esta operación, con el proposito de tener esquemas mucho más eficientes. La multiplicación escalar es eficiente si la aritmética en el cuerpo es eficiente, por esa razón proponer cuerpos de extensión óptimo para curvas elípticas y computar la multiplicación escalar es otro motivo para comparar estos tres tipos de cuerpos y así proponer a los CEO como otra alternativa viable en este tipo de estudio.

A continuación se describen las implementaciones y resultados que muestran el rendimiento de las operaciones en un cuerpo de extensión óptimo en particular. Con el proposito de mostrar el rendimiento de la multiplicación escalar de una curva elíptica definida sobre este cuerpo.

4.2. Operaciones módulo $p = 2^{54} - 33$

En esta sección, se presentan algunas tablas que describen el rendimiento de las implementaciones de las operaciones reducción, multiplicación y división entera, multiplicación e inversión de polinomios y exponenciación cuadrada en el CEO $\mathbb{F}_{p^3}$ con $p = 2^{54} - 33$, definido por el binomio irreducible $x^3 - 2$

Las unidades FFs, LUTs, Slices son los recursos lógicos que posee el FPGA y son agrupados en "slices"para crear un bloque lógico configurable. Estos "slices" contienen un número de LUTs, flip-flops y multiplicadores. Un LUT es una colección de puertos lógicos cableados en un FPGA, estos almacenan una lista predefinida de salidas para cada combinacion de entradas, los cuales proporcionan una manera rápida de recuperar el resultado de una operación lógica. Un Flip-Flop es un pequeño circuito capaz de tener dos estados estables.

Las tablas que se presentan a continuación indican los recursos lógicos de consumo en el FPGA para cada sintesis de los programas. Además se indica el promedio de los ciclos y períodos de tiempo que nos serviran para computar un tiempo promedio de cada simulaciónES de los programas. Note que el tiempo total es computado por, $t = (\#ciclos) * (periodo)$ y que las unidades están dadas en nanosegundos(ns).

4.2.1. Reducción módulo $2^{54} - 33$

En el capítulo 2, en la sección de inversión directa en un cuerpo de extensión óptimo, se necesita computar el siguiente valor $\Delta = a_0^3 - 3a_0a_1a_2\omega + a_1^3\omega + a_2^3\omega^2$ para caso especial $m = 3$ y el binomio irreducible $x^3 - \omega$. Como nuestro caso también $m = 3$ y nuestro objetivo es hacer la menor cantidad de reduciones módulo p, debemos entonces buscar el mayor elemento posible a reducir, en este caso Δ.
Dado que nuestro binomio es de la forma x^3-2 entonces $\Delta = a_0^3-6a_0a_1a_2+2a_1^3+4a_2^3$ y como a_0, a_1, a_2 están en $\mathbb{F}_p$ si tomamos $a = \max\{a_0, a_1, a_2\}$ y $b = \min\{a_0, a_1, a_2\}$ se puede probar que $\Delta < 6(a-b)^3$ por tanto Δ a lo más, es de 165 "bits".

El siguiente procedimiento muestra como reducir un número x de 165 "bits" módulo $p = 2^{54} - 33$.

$$\begin{aligned} x &= x_{164}2^{164} + \cdots + x_{108}2^{108} + x_{107}2^{107} + \cdots + x_{54}2^{54} + x_{53}2^{53} + \cdots + x_0 \\ &= (x_{164}2^{56} + \cdots + x_{108})2^{108} + (x_{107}2^{53} + \cdots + x_{54})2^{54} + x_{53}2^{53} + \cdots + x_0 \end{aligned}$$

como $2^{54} \equiv 2^5 + 1 \bmod p$, entonces $2^{108} = (2^{54})^2 \equiv (2^5 + 1)^2 = 2^{10} + 2^6 + 1 \bmod p$ luego,

$$x \equiv (x_{164}2^{56}+\cdots+x_{108})(2^{10}+2^6+1)+(x_{107}2^{53}+\cdots+x_{54})(2^5+1)+x_{53}2^{53}+\cdots+x_0 \bmod p$$

El producto $(x_{164}2^{56} + \cdots + x_{108})(2^{10} + 2^6 + 1)$ se puede computar trasladando los "bits" $x_{164} \cdots x_{108}$ hacia la izquierda un total de 16 posiciones y luego 2 adiciones. Similarmente el producto $(x_{107}2^{53}+\cdots+x_{54})(2^5+1)$ se puede computar trasladando los "bits" $x_{107} \cdots x_{54}$ un total de 5 posiciones hacia la izquierda y luego haciendo una adición.

Resulta entonces que $x \equiv x' \bmod p$ donde

$$x' = x'_{66}2^{66} + \cdots + x'_{54}2^{54} + x'_{53}2^{53} + \cdots + x_0$$

consiste de 67 "bits" pero

$$\begin{aligned} x' &= \left(x'_{66}2^{12} + \cdots + x'_{54}\right)2^{54} + x'_{53}2^{53} + \cdots + x_0 \\ &\equiv \left(x'_{66}2^{12} + \cdots + x'_{54}\right)(2^5 + 1) + x'_{53}2^{53} + \cdots + x_0 \bmod p \end{aligned}$$

y después de simplificar esto nos da una nueva expresión x'' que tiene a lo más 54 "bits" lo que implica que $x \bmod p$ es x'' ó $x'' - p$

Note que para un primo $p = 2^n - c$, la forma de c es muy importante al momento de hacer reducción módulo p. El hecho de que en este caso $c = 33$ sea de la forma

$c = 2^5 + 1$ ayuda a que la reducción módulo p pueda ser llevada a cabo con un mínimo número de desplazamientos y sumas, a diferencia del hecho de que no hubiera tenido una forma peculiar. Si este hubiera sido el caso, no habría más camino que simplemente multiplicar c de forma ordinaria al momento de hacer el procedimiento anterior y esto podría resultar muy costoso a nivel de la implementación, ya que este es un componente que se necesitará reutilizar muchas veces, cuando se necesite reducir elementos al cuerpo base.

Los resultados de la implementación son mostrados en la siguiente tabla

	FFs	**LUTs**	**Slices**	**Tiempo (ns)**
Reductor54b	438	465	313	8

Tabla 4–1: Reducción módulo $p = 2^{54} - 33$

Veamos a continuación las operaciones de multiplicación y reducción el binomio irreducible.//

4.2.2. Multiplicación y reducción en $\mathbb{F}_{p^3}$

En el capítulo 2, se describe que la multiplicación en cuerpos de extensión óptimo puede ser representada vía matricial a partir de la matriz de Mastrovito modificada. Se aclara también, el hecho de que es necesario generar solo la primera fila dado que cada fila despúes de la primera puede ser generada multiplicando el primer elemento de la fila actual por ω y luego haciendo un desplazamiento circular hacia la izquierda.

En nuestro caso, dado que $m = 3$ es un valor bastante pequeño, no será necesario hacer los desplazamientos circulares. El hecho de tener $m = 3$ es muy útil, ya que solo debemos hallar los valores de c_2, c_1, c_0 que se describen en la representación matricial, esto es

$$\begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} a_2 & a_1 & a_0 \\ a_1 & a_0 & 2a_2 \\ a_0 & 2a_2 & 2a_1 \end{bmatrix} \cdot \begin{bmatrix} b_0 \\ b_1 \\ b_2 \end{bmatrix}$$

por tanto,

$$c_2 = a_2b_0 + a_1b_1 + a_0b_2 \tag{4.1}$$

$$c_1 = a_1b_0 + a_0b_1 + 2a_2b_2 \tag{4.2}$$

$$c_0 = a_0b_0 + 2a_2b_1 + 2a_1b_2 \tag{4.3}$$

Otra ventaja, para la implementación es que $\omega = 2$, hecho que nos indica que solo debemos hacer un desplazamiento hacia la izquierda en los términos donde se multiplica por 2. Observe que cada c_i cumple que $c_i \leq 5(p-1)^2$ es decir que cada c_i tiene a lo más 111 "bits", luego el reductor descrito anteriormente funciona para cada c_i.

Los resultados de la implementación son mostrados en la siguiente tabla.

	FFs	**LUTs**	**Slices**	**Prom. ciclos**	**Periodo (ns)**	**Prom. Tiempo (ns)**
Multpol54b 1	1994	4104	2191	55	1	55
Multpol54b 2	1101	2942	1329	3	2	6

Tabla 4–2: Multiplicación en $\mathbb{F}_{p^3}$

Para la implementación fueron diseñados dos versiones, el primero depende de la señal de reloj y el segundo usa bucles "for", sin embargo a pesar de que el segundo multiplicador es mucho más rápido que el primero, para la implementación final de la multiplicación escalar en curvas elípticas, se utilizó el primero debido a que este se adaptó mejor al diseño general de los demas algoritmos.

Por otro lado, en [7] se describen los siguientes resultados para multiplicadores en el cuerpo $\mathbb{F}_{2^{163}}$ módulo el polinommio irreducible $x^{163} + x^7 + x^6 + x^3 + 1$.

Como se puede observar, Multpol54b 1 es más rápido comparado con Interleaved 1 y Montgomery, sin embargo ambos consumen menos área. Con respecto a

	FFs	LUTs	Slices	Prom. ciclos	Periodo (ns)	Prom. Tiempo (ns)
Interleaved 1	534	2237	1190	11	5.9	65
Interleaved 2	589	6956	3588	3	9.7	29
Montgomery	344	347	184	163	7.4	1206

Tabla 4–3: Multiplicadores en $\mathbb{F}_{2^{163}}$

Interleaved 2, Multpol54b 1 es más lento, pero nuestro multiplicador consume menos LUTs. Multpol54b 2 es más rápido en cualquier caso y consume menos área (excepto por los FFs) que interleaved 2.

Veamos a continuación las operaciones de división entera módulo p e inversión en el CEO.

4.2.3. División e inversión en $\mathbb{F}_{p^3}$

En el capítulo 2, se menciona que la división en el cuerpo base $\mathbb{F}_p$ puede ser computada mediante el algoritmo extendido de Euclides. Sin embargo, este método resulta costoso a nivel de hardware, por el hecho de utilizar muchas divisiones repetivas. Por esa razón, se presenta otra alternativa que es el algoritmo binario para la división modular, que recibe su nombre debido a las divisiones por 2, que hay que relizar en el procedimiento.
Los resultados de la implementación son mostrados en la siguiente tabla

	FFs	LUTs	Slices	Prom.ciclos	Periodo (ns)	Prom. Tiempo (ns)
division54b	218	1044	545	58.5	2	117

Tabla 4–4: División módulo $p = 2^{54} - 33$

Por otro lado, el inverso de un elemento $a(x)$ en $\mathbb{F}_{p^3}$ es otro polinomio $b(x)$ tal que $a(x)b(x) \equiv 1 \bmod x^3 - 2$. En el capítulo 2, se describe la inversión directa para casos particulares como $m = 2$ y $m = 3$ como es nuestro caso. Dado que $\omega = 2$ se

tiene que las formulas para la inversión directa en el caso $m = 3$ son:

$$\begin{aligned} b_0 &= (a_0^2 - 2a_1a_2)\Delta^{-1}, \\ b_1 &= (2a_2^2 - a_0a_1)\Delta^{-1}, \\ b_2 &= (2a_1^2 - a_0a_2)\Delta^{-1}, \end{aligned}$$

donde $\Delta = a_0^3 - 6a_0a_1a_2 + 2a_1^3 + 4a_2^3$.

Nuevamente al igual que en la multiplicación, la inversión nos indica que solo debemos hacer un desplazamiento hacia la izquierda en los términos donde se multiplica por 2. Tenemos que cada b_i se peude reexpresar como

$$\begin{aligned} b_0 &= (a_0^2 + 2(p - a_1)a_2)\Delta^{-1}, \\ b_1 &= (2a_2^2 + (p - a_1)a_0)\Delta^{-1}, \\ b_2 &= (2a_1^2 + (p - a_2)a_0)\Delta^{-1}, \end{aligned}$$

esto lo hacemos para evitar restar, como mostraba la fórmula anterior, lo que implica que cada $b_i \leq 3(p-1)^2$ es decir que cada b_i tiene también a lo más 110 "bits", luego el reductor descrito anteriormente funciona para cada b_i, asi como para Δ como se detallo anteriormente.

Los resultados de la implementación son mostrados en la siguiente tabla

	FFs	**LUTs**	**Slices**	**Prom. ciclos**	**Periodo (ns)**	**Prom. Tiempo (ns)**
inverseoef54b	1318	12117	3196	218.4	2	436.8

Tabla 4–5: Inversión en el cuerpo $\mathbb{F}_{p^3}$

En [7], al igual que para la multiplicación, encontramos la siguiente tabla de resultados para la división de polinomios en $\mathbb{F}_{2^{163}}$,

	FFs	LUTs	Slices	Prom. ciclos	Periodo (ns)	Prom. Tiempo (ns)
divisionK163	679	726	544	326	7.4	2413

Tabla 4–6: División en el cuerpo $\mathbb{F}_{2^{163}}$

Esto nos indica que para invertir un polinomio en $\mathbb{F}_{2^{163}}$ en promedio se tarda en 5.5 veces más, que invertir un polinomio en $\mathbb{F}_{p^3}$, sin embargo el consumo en área para invertir en el cuerpo binario es mucho menor que el CEO.

A continuación se describe como calcular cuadrados en el cuerpo $\mathbb{F}_{p^3}$.

4.2.4. Exponenciación cuadrada en el cuerpo $\mathbb{F}_{p^3}$

La exponenciación cuadrada debe ser implementada usando el método para la multiplicación general explicado en el capítulo 2. Además, observaremos que el elevar al cuadrado un elemento del cuerpo permite algunas eficiencias computacionales adicionales. Por ejemplo, consideremos el elemento del cuerpo $a(x) = a_2x^2 + a_1x + a_0,\ a(x) \in \mathbb{F}_{p^3}$. Se calcula el cuadrado de $A(x)$ y se obtiene:

$$(a_2x^2 + a_1x + a_0)^2 = a_2^2x^4 + 2a_2a_1x^3 + [2a_2a_0 + a_1^2]x^2 + 2a_1a_0x + a_0^2$$

Los resultados de la implementación son mostrados en la siguiente tabla

	FFs	LUTs	Slices	Prom. ciclos	Periodo (ns)	Prom. Tiempo (ns)
cuaoef54b	1329	4073	2174	55	1	55

Tabla 4–7: Exponenciación cuadrada en el cuerpo $\mathbb{F}_{p^3}$

También en [7] presentan resultados acerca de la exponenciación cuadrada en $\mathbb{F}_{2^{163}}$, en general esta implementación es mucho más rápida y consume menos área que la del cuerpo de extensión óptimo $\mathbb{F}_{p^3}$, la tabla siguiente muestra los resultados

	LUTs	Slices	Tiempo
squaring	165	86	3

Tabla 4–8: Exponenciación cuadrada en el cuerpo $\mathbb{F}_{2^{163}}$

4.3. Multiplicación escalar en curvas elípticas sobre $\mathbb{F}_{(2^{54}-33)^3}$

En el capítulo 3, se menciona que la multiplicación de dos puntos en un grupo definido por una curva elíptica no existe. Sin embargo, la multiplicación escalar kP donde k es un entero positivo si se puede calcular.

El método descrito en el capítulo 3, es el método binario, que recibe su nombre dado que se hace uso de la representación binaria del valor de k.

Los datos y resultados presentados a continuación se hicieron bajo una curva definida sobre el cuerpo $\mathbb{F}_{p^3}$ con $p = 2^{54} - 33$ y binomio irreducible $x^3 - 2$.

La curva es definida como:

$$a \;:\; \texttt{005594eac3b3b0 2d34732b5a492e 2612dca8db442b}$$

$$b \;:\; \texttt{2db84cd8217011 0cdecb4eeeaa71 27dd891a598787}$$

$$E \;:\; y^2 = x^2 + ax + b$$

usando una notación hexadecimal. Y el punto elegido $P \in E$

$$P_x \;:\; \texttt{3c40da2e0ecd7b 10d1bfbbc1177e 206bce49cf6243}$$

$$P_y \;:\; \texttt{2904fdf26e5fc6 3fa0afeacfcd78 0eb04ac11014fc}$$

Los valores de kP computados en la implementacion, fueron para valores de k aleatorios para un intervalo de 1 hasta 155 bits.

Los resultados de la implementación son mostrados en la siguiente tabla

	FFs	LUTs	Slices	Prom. ciclos	Periodo (ns)	Prom. Tiempo (ns)
multescalar54b	18262	60101	32217	58195.5	1.3	75654.15

Tabla 4–9: Multiplicación escalar sobre cuerpo $E(\mathbb{F}_{p^3})$

Por otro lado, los resultados mostrados en [7], muestran el costo y retraso para la multiplicación escalar de una curva definida en $E(\mathbb{F}_{2^{163}})$.

	FFs	LUTs	Slices	Prom. ciclos	Periodo (ns)	Prom. Tiempo (ns)
pointmultiplicationK163	2170	3514	2062	54422.8	7.9	429940

Tabla 4–10: Multiplicación escalar sobre cuerpo $E(\mathbb{F}_{2^{163}})$

Como se puede observar, nuestro multiplicador es hasta 4 veces más rápido que la versión en [7], sin embargo el consumo en área si es mucho mayor que la versión en [7], una explicación a esto puede ser la reutilización de código que hace que unidades como los slices y FFs crezcan , sin embargo como nuestro objetivo era tener obtener un buen tiempo, el evitar la reutilización de código implicaba hacer mayor sincronización en las señales para el diseño en nuestro circuito y esta sincronización generaba más rertraso en el sistema.

Capítulo 5
CONCLUSIONES Y TRABAJOS FUTUROS

Los resultados anteriores muestran que nuestra implementación de la multiplicación escalar para una curva E definida sobre $\mathbb{F}_{(2^{54}-33)^3}$ es en promedio cuatro veces más rápido que la multiplicación escalar en $E(\mathbb{F}_{2^{163}})$. Esto nos lleva a pensar que si es posible encontrar variantes de CEO $\mathbb{F}_{p^m}$ con $p = 2^n - (2^i + 1)$ y $x^m - 2$ irreducible, que optimicen el rendimiento, por lo menos en tiempo, para las operaciones de curva eíptica.

Sin embargo, hay que anotar que el consumo en área para el cuerpo $E(\mathbb{F}_{2^{163}})$ fue significativamente menor al del CEO escogido. Una de las razones por la que se pudo haber consumido más área para el caso $\mathbb{F}_{(2^{54}-33)^3}$ es la reutilización de código en la implementación. Mejorar este punto podría significar un motivo más para decidir entre la escogencia de este tipo de cuerpos. Así mismo, intentar usar otro tipo de multiplicadores sobre $\mathbb{F}_p$ distintos al clásico podría motivar a nuevas comparaciones que permitan determinar cual multiplicador es más apropiado.

Por otro lado, establecer condiciones necesarias y suficientes para que dado un número positivo q exista un CEOFPGA, podría ayudar a establecer nuevas alternativas y estándares en criptografía. Así mismo, investigar otros casos para m distinto de 3 e investigar algunas otras aplicaciones en las que puedan ser de utilidad el uso de CEOFPGA, como por ejemplo encriptación de imagenes.

Bibliografía

[1] L-M. Adleman ,R-L Rivest y A. Shamir. *"A Method for obtaining Signatures and Public Key Criptosystems"*. Communications of the ACM, vol 21, 120-126, 1978.

[2] S. Amara y A. Moncef, "*Hardware Implementation of Elliptic Curve Point Multiplication over $GF(2^m)$ for ECC protocols"*. IEEE Trans. Inform. Theory, vol 44: 16-31, 1998.

[3] H. Baier, "*Elliptic Curves of Prime Order over Optimal Extension Fields for Use in Cryptography"*. Springer Berlin Heidelberg, 2001.

[4] D. Bailey y C. Paar, "*Optimal Extension Fields for Fast Arithmetic in Public-Key Algorithms"*. Springer-Verlag, 472-485, 1998.

[5] S. Baktir y B. Sunar, *"Optimal Tower Fields"*. IEEE Trans. Comp, vol 53, Oct. 2004.

[6] P. Chu Pong, *"RTL Hardware Design Using VHDL"*. Wiley-Interscience, 2006.

[7] J-P. Deschamps, G. Sutter y J. Imaña, *Hardware Implementation of Finite Fields Arithmetic* . McGraw Hill, 2009.

[8] W. Diffie y M.E. Hellman. *"New Directions in Cryptography"*. IEEE Trans. Inform. Theory, 11-22, 644-654, 1976.

[9] T. ElGamal, *"A Public Key Criptosystem and a signature scheme based on discrete Loarithms"*, IEEE Trans. Inform. Theory, vol 31, 469-472, 1985.

[10] E. Ferrer , D. Bollman, O. Moreno, *"A Fast Finite Field Multiplier"*, Lecture Notes in Computer Science, Springer, 4419, 238-246, 2007.

[11] Hankerson, Menezes, Vanstone. 2004. "*Guide to Elliptic Curve Cryptography"*. Springer.

[12] Hungerford Thomas. *"Algebra"*. Springer, 2000.

[13] N. Koblitz. *" Elliptic Curve Criptosistems"*. Mathematics of Computation, vol 48, 203-209, 1987.

[14] LiDIA. *" A C++ Library For Computational Number Theory "*. http://www.cs.sunysb.edu/~algorith/implement/lidia/implement.shtml

[15] R. Lidl y H. Niederreiter. *"Finite Fields"*. Cambridge University Press, 2da edición, 1997.

[16] E. Morales, "*Criptografía de curva elíptica sobre campos finitos y su aplicación al cifrado de imagenes"*. Trabajo de Grado. Universidad del Atlántico, Colombia, Marzo 2010.

[17] E. Morales, "*An FPGA Implementation of Fast Finite Field Multiplication*". Póster. NEA Science Day - UPRM. Puerto Rico, Mayagüez. 2013.

[18] E. Morales y M. Gonzalez, "*An FPGA Implementation of Fast Finite Field Multiplication"*. Póster. SACNAS. Seattle, WA, Octubre 2012.

[19] A. Mahouz y G. Hancke, "*An Efficient Method for Finding Square Roots for Elliptic Curves over OEF"*. FCS, 87-91, 2009.

[20] V. Miller, *" Use of Elliptic Curve in Cryptography"*. CRYPTO, 417-426, 1985.

[21] National Institute of Standards and Technology. Julio 1999. http://csrc.nist.gov/groups/ST/toolkit/documents/dss/NISTReCur.pdf

[22] V. Pedroni, *"Circuit Design with VHDL"*. MIT Press. 2004.

[23] F. Rodriguez , N. Saqib, A. Díaz, C. Kaya ,. *"Cryptographic Algorithms on Reconfigurable Hardware"*. Springer, 2006.

[24] SAGEMATH. v5.9 (2013-04-30). http://www.sagemath.org/

[25] Standards for Efficient Cryptography Group. 2009."*SEC 2: Recommended Elliptic Curve Domain Parameters"*. Certicom Corp.

[26] L. Washintong, "*Elliptic Curves Numbers Theory and Cryptography"*. CHAPMAN and HALL/CRC, 2003.

[27] A. Woodbury , D. Bailey y C. Paar, "*Elliptic Curve Cryptography On Smart Cards Without Coprocessors*". Kluer. 71-92, 2000.